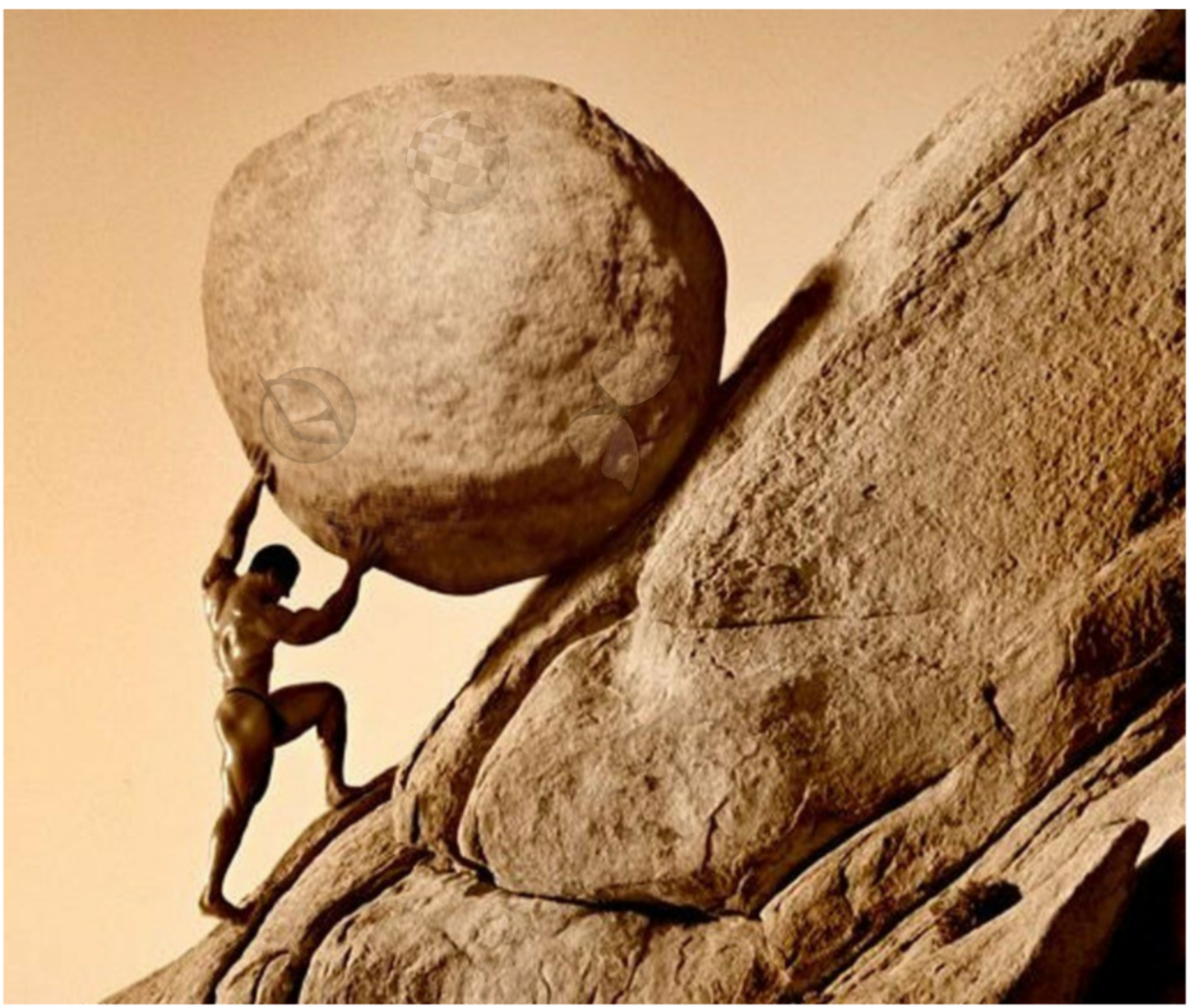
AMIGA NG

AMIGA OS 4 - MORPH OS - AROS 01/2017



WSTĘPNIAK

Chciałbym napisać “trzymasz w ręku pierwszy numer”, ale niestety, nie tym razem. Startujemy od wersji elektronicznej jedynie. Może skromnie objętościowo, ale mam nadzieję że różnorodność tematów to rekompensuje.
Plany są większe, ale od czegoś zacząć trzeba.

Czasopismo chciałbym żeby miało cykl kwartalnika. Profil - szeroko rozumiana “Amiga NG” (AmigaOS 4, MorphOS, AROS). Nie ma takiego pisma na rynku, chcę wypełnić tę lukę.

W tym numerze nie ma działów (za mała objętość całości), ale chcemy wprowadzić stałe elementy (publicystyka, wywiad, listy od czytelników). Także wypłynąć na szerokie wody (wersja angielska, druk). Jest dużo pomysłów, ale do tego potrzeba więcej osób.
“Amiga NG” jest częścią projektu amiga.org.pl - tam znajdują się warunki współpracy i numery do pobrania.

Dziękuję wszystkim, którzy dali namówić się na udział, dla niektórych to pierwsze artykuły w życiu. Taki też był pośredni cel, uaktywnić osoby które do tej pory nie udzielały się pisarsko.
Mam nadzieję, że spotkamy się za trzy miesiące w większym gronie,
i będziemy rozwijać nasze hobby wspólnie.

Adam Mierzwa

Spis treści

Nowości 2
Wybrane wydarzenia

Podmiana obrazka startowego w OS4 4
Jak złamałem plik BootImage z Kickstartu

Aleph One / Marathon 8
Znana trylogia FPP z Macintosha pod OS4

Hollywood #1 - Tagać, tagać, tagać... 11
Początki pracy z programem, przede wszystkim o fontach

Konfiguracja Lightwave 20
Ustawianie programu pod MorphOS

Kotek na motylku 23
Emulacja konsoli Jaguar pod MorphOS

Mapparium 27
Przeglądarka Open Street Map dla systemu AROS

Xgalaga 29
Kosmiczna strzelanka dla AROS-a

Pomidor 31
O zwiększaniu wydajności pracy

AMIGA NG

Numer 1/2017

Redakcja:

Adam Mierzwa
adammierzwa1@gmail.com

Autorzy tekstów:

Radosław Czernik
Waldemar Karaś
Robert Krajcarz
Adam Mierzwa
Rafał Pączko

Skład: PagePlus X9

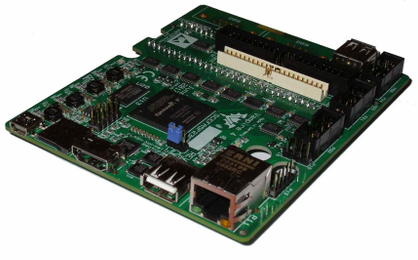
Nowości

Jako że to pierwszy numer i nie ma odniesienia - wybór tego co wydarzyło się w ostatnim kwartale. Następnym razem będzie już to co wydarzyło się pomiędzy kolejnymi wydaniami. Szczerze powiedziawszy marny okres, ale tradycyjnie najlepsze rzeczy wypływają końcem roku.

Vampire 4

Zespół Apollo zapowiedział wydanie pod koniec roku nowej wersji Vampire w trzech wersjach:
- dla A1000, 500, 2000 i 600 (z adapterem)
- dla A1200
- wersja samodzielna

Ta ostatnia szczególnie wzbudza emocje. Płytka będzie oparta o układ Altera Cyclone V A5, mieć 512 MB DDR3 RAM, implementację procesora 68080 oraz chipset SAGA (kompatybilny z AGA),
Będą złącza FastIDE 40 oraz 44 pin, wyjście wideo 720p, podwójny flashrom na kickstart, slot kart pamięci MicroSD, dwa porty USB oraz karta sieciowa.

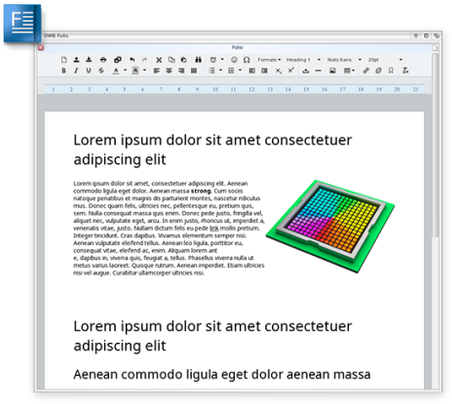


Na zdjęciu Vampire V4, PCB rev. 3B

Folio

Prosty procesor tekstu, powstały z inicjatywy użytkowników MorphOS. Napisany w języku JavaScript, działa w przeglądarce internetowej.

Jego zaletą jest to, że pracuje w trybie offline.



Więcej o Folio - tutaj:
www.amigaimpact.org/2017/08/mos-folio-tourne-la-page-et-passe-en-beta/

Amiparty XXI

W dniach 11-13 sierpnia odbyło się 21 edycja AmiParty. W tym roku zawitali wyjątkowi goście - Trevor Dickinson i David Pleasance.
Impreza była w zgodnej opinii bardzo udana.
Krótka relacja i linki do galerii - na blogu Dickinsona:
blog.a-eon.biz/blog/?m=201709

BlackBOX 1200

Nowa partia w sprzedaży od sierpnia 2017 r. Produkt wprowadzony przez rodzimy Elbox w kwietniu, jest przeznaczony dla Amigi 1200. Umożliwia wygodną rozbudowę (Mediator PCI, i t.d.).
Bazą jest obudowa Zalman R1. Udogodnienia to przede wszystkim zmieniony tylni panel.

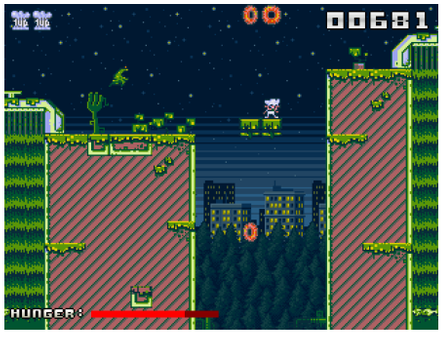
Zainstalowany jest zasilacz 350 W (wystarczy do zasilenia najbardziej rozbudowanych A1200), Dołączony jest interfejs klawiatury, panel frontowy ma już zabudowany front do stacji FDD montowanych w A1200, a w górnym panelu zamontowany jest odpowiedni układ sterowania LEDów.
Cena to 299,95 euro.

Pełna specyfikacja i możliwość zakupu pod tym adresem:
eu-shop.elbox.com/en_GB/p/BlackBOX-1200/734

Więcej zdjęć znajduje się w instrukcji montażu:
elbox.com/ma/b-box_1200_folder/b-box_1200.html

Hermes

Niezła platformówka stworzona przez grupę Retroguru.
Dostępna na prawie wszystkie platformy (także nieamigowe).



Grę można pobrać na stronie projektu:
www.retroguru.com/hermes/

AmigaOne w reklamie

Komputer X5000 w obudowie z X1000 "zagrał" w reklamie amerykańskiej sieci salonów fryzjerskich. Fakt ten jest pozornie nieistotny, to precedens - komercyjne lokowanie produktu.

Klip można obejrzeć pod adresem:
vimeo.com/230639859

Phase 5

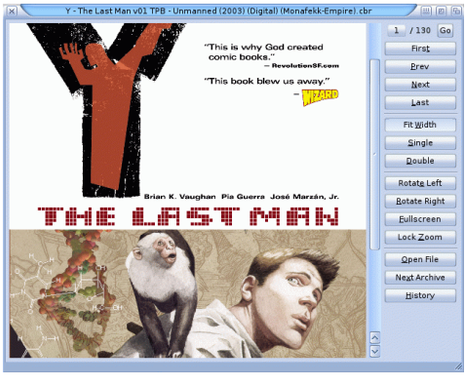
Uznawana przez wielu za mistyfikację, reaktywowana Phase 5, przedstawiła cennik:

Fizyczny, działający produkt ma być dostępny jeszcze w tym roku.

Zainteresowani powinni śledzić profil publiczny Phase 5 na fb, który jest źródłem informacji "z pierwszej ręki":
pl-pl.facebook.com/phase5digital/

RNO Comics

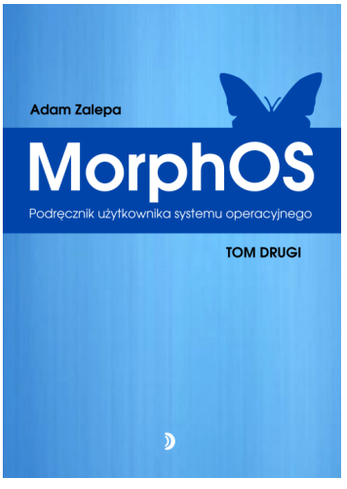
Przeglądarka komiksów (wsparcie dla formatów CBR, CBZ, CBT).
Dostępna dla MorphOS i AmigaOS 4.



Strona programu:
http://jpv.wmhost.com/jpv_software/RNOComics/

Książka o MorphOS

Zapowiedziano drugi tom o tym systemie. Ma on ukazać się w pierwszym kwartale 2018 r. Za przedsięwzięcie odpowiedzialne jest wydawnictwo A2.
Bieżących informacji należy wypatrywać pod tym adresem:
amiga.net.pl/index.php?lng=pl&kwt=10010&tms=204



AmiWest 2017

W dniach 19-22 października odbędzie się najważniejsza impreza dla środowiska OS 4 czyli AmiWest.

Plan imprezy, wystawcy i bieżące informacje - na oficjalnej stronie:
www.amiwest.net

Podmiana obrazka startowego w OS4

Phibrizzo

W trakcie startu systemu AmigaOS4.1 na ekranie pokazuje się obrazek symbolizujący z czym mamy do czynienia. Po jakimś czasie zaczyna on nam się nudzić i najchętniej chcielibyśmy go zmienić na coś swojego, indywidualnego.

Jakiś czas po tym jak na swoim SAM440ep zainstalowałem OS4.1, przyszedł mi do głowy pomysł żeby podmienić oryginalny obrazek startowy na jakiś taki który by mi bardziej odpowiadał. Niestety, choć przeszukałem wszystkie opcje w preferencjach Workbencha nie znalazłem żadnej która by to umożliwiała.
Zapytałem więc na forum eXeca czy jest to w ogóle możliwe. W odpowiedzi dowiedziałem się że obrazek startowy który pokazuje się na starcie systemu jest zawarty w jednym z plików KickStartu.
W związku z tym podmiana go w sposób bezpośredni nie jest w żaden sposób możliwa.
Szaman tak mimochodem napisał: "A może byś spróbował?". Jako że za młodu bawiłem się w ripowanie modułów z gier i programów demonstracyjnych więc czemu nie. Zabrałem się za to.

Wszystkie opisane tu czynności wykonywałem na pliku BootImage z systemu AmigaOS4.1 który był dostarczany przy zakupie komputera SAM 440ep. Jest to wersja starsza niż Final Edition. Ponieważ nie posiadam tej wersji nie jestem w stanie stwierdzić czy operacje które tu opiszę będą w dalszym ciągu aktualne.
Jednak do Update 6 wszystko działa bez zarzutu.

Na samym początku trzeba się dowiedzieć w którym konkretnie elemencie Kickstartu może znajdować się ten obrazek. W systemie 4.x KickStart nie jest umieszczony w pamięci ROM jak to ma miejsce starszych systemach. Dodatkowo nie jest też w jednym kawałku lecz składa się kilkunastu pojedynczych plików które w trakcie startu systemu są wczytywane z twardego dysku. Ma to swoje zalety gdyż jakakolwiek aktualizacja KickStartu polega tylko na podmianie konkretnego pliku. Taka sytuacja sprawę podmiany obrazka bardzo upraszcza.

Odszukanie konkretnego pliku było banalnie proste. Wystarczyło wejść do katalogu Sys:Kickstart.
Plik którego szukałem sam swoją nazwa BootImage podpowiadał że jest tym czego szukałem.

Niestety. Plik ten nie był jakimś zwykłym obrazkiem w jednym z popularnych formatów ze zmienioną nazwą lecz programem który posiada w sobie dane tego obrazka zapisane tylko w sobie znanym formacie i wyświetla w odpowiednim momencie.
Dekompilacja tego programu w ogóle nie wchodziła w grę. Każdy z czytających który spodziewał się profesjonalnego podejścia do tematu z pokazaniem jak to można prosto zrobić dostępnymi programami muszę rozczarować. Opisane przeze mnie czynności można potraktować jako prymitywny włam w zawartość pliku wykorzystując do tego własne oprogramowanie.

ETAP PIERWSZY - Obrazku pokaż się!

W pierwszej kolejności należało ustalić w jakiej wielkości i w ilu kolorach mógłby to być obrazek.
Monitor do którego miałem podłączony komputer wykazał że aktualnie wyświetlany obraz ma rozmiary 640x480.
To już jest jakaś wskazówka. Szybka kalkulacja wykazała jednak że wielkość obrazka znacznie przekracza wielkość pliku.
A więc obrazek musi być jakoś skompresowany a to z kolei może utrudnić sprawę.

Dalsze czynności nad plikiem przeprowadzałem na mojej Amidze 1200 gdzie mam środowisko programistyczne.
Wykorzystując File Mastera, który posiada opcje hex-edytora, postanowiłem poszukać w zawartości pliku slow kluczowych takich jak: PNG, JPG, JFIF, ILBM i t.p,

w nadziei że zapisany obrazek jest tam po prostu doklejony.
Poszukiwania niestety niczego nie przyniosły.

Należało mu się więc baczniej przyjrzeć w poszukiwaniu czegokolwiek co mogło by przypominać dane graficzne.

W trakcie jego przeglądania w oczy rzucił mi się offset 0x010058 (65624) w którym dane podejrzanie się powtarzały.
Przypominało to sposób zapisu palety barw w systemie amigowym, gdzie dane koloru jednego pisaka w trybie 8-bitowym zapisane są w trzech długich słowach (3x32 bity). Jednak po przeliczeniu ilości tak zapisanych barw wychodziło 3036 bajtów zamiast oczekiwanych 3072 co daje tylko 253 barwy. Na szczęście mówił o tym offset 0x010055 (65621). Po palecie barw pod offsetami 0x010C38 (68664), 0x010C3A (68666) znajdowały się dane 0x0280, 0x01E0 czyli w przeliczeniu na układ dziesiętny odpowiednio 640 oraz 480. To mogła być wielkość obrazka a wiec analiza pliku szła w dobrym kierunku.
Za tymi danymi od offsetu 0x010C30 (68668) następowały kolejne nic już w sumie nie mówiące dane.
Skopiowałem te dane programem który napisałem bardzo dawno temu którego używam do dziś.
Pozwala on na skopiowanie dowolnej długości danych z danego pliku zaczynając od dowolnego miejsca.
Nazwałem go sobie Kopier. Programy tego typu prawdopodobnie można znaleźć na Aminecie.

Analiza wycinka wykazała że dane były skompresowane metoda RLE (Run-Length Encoding). W skrócie jest to metoda bezstratnej kompresji, która polega na tym że ciągi danych następujących po sobie które się powtarzają zastępuje się jednym plus informacje ile razy taki element się powtórzył. Elementy niepowtarzające się przepisuje się.
Dla przykładu: "AAAMIGAAA RULEZZZZ" można zapisać jako: "3A3MIG3A5 RULE4Z".
Opis troszeczkę uprościłem gdyż w rzeczywistości trzeba rozróżnić które dane się powtarzają a które nie.
Wykorzystano tutaj znak liczby informacyjnej: jeśli jest dodatni to elementy nie powtarzają się, jeśli ujemny - powtarzają. Algorytm prymitywny ale w opisywanym przypadku wystarczający.
Osobom trochę bardziej obeznanym z Amigą przypomina to kompresję stosowaną w plikach ILBM. I tak w istocie jest.
Tyle tylko że w omawianym przypadku dane o samym obrazku są inaczej opisane, dane jednego piksela opisuje jeden bajt.

Wiedząc o tym, szybko napisałem w C program który zdekompresował mi te dane. Ilość informacji po dekompresji ilościowo zgadzała się z zakładaną jaka powinna być dla obrazka o rozmiarach 640 na 480 czyli 327200 bajtów.
Należało je tylko w jakiś sposób zweryfikować. Tutaj z pomocą przyszedł mi program Personal Paint i format graficzny BMP.
Format ten zapisuje dane wprost bez kompresji z tą różnicą że linie obrazu zapisuje zaczynając od ostatniej.
Jednak dla testu nie było to problemem. Stworzyłem w PPaincie pusty obrazek o wymiarach 640x480 z paletą jaka była ustawiona na starcie programu i zapisałem jako BMP.

Wiedząc że dane graficzne w takim pliku zaczynają się od offsetu 0x0436 (1078) aż do końca pliku, podmieniłem je poprzez odpowiednie pocięcie wyżej opisanym programem i doklejenie swoich danych poleceniem Join. Ponowne załadowanie takiego pliku do PPainta i... BINGO!

Oryginalny obraz wydobyty z BootImage

Moim oczom ukazał się dobrze znany obrazek tyle że obrócony do góry nogami z niepoprawną paletą.

Teraz wszystko poszło z górki. Należało jeszcze podmienić paletę która oryginalnie była w 96 bitach na 0RBB (32 bity) gdyż taki zapis obowiązuje w formacie BMP. Do tego także napisałem własny program który mi to przekonwertował.
Paleta w pliku BMP zapisana jest od offsetu 0x0036 (54), wiedząc to wykonałem podmianę danych i ostatecznie dostałem to co liczyłem. Wystarczyło tylko obrócić obrazek wczytany do PPainta (ilustracja na poprzedniej stronie).

Podsumuję czego się dowiedziałem:
- obrazek ma rozmiar 640x480 w 8 bitach
- dane obrazka spakowane są metodą RLE
- offset gdzie zaczynają się dane obrazka to 0x010C30 (68668)
- całkowita długość spakowanych danych oryginalnego obrazka to 69741 bajtów
- paleta obrazka zapisana jest w 12 bajtach na kolor
- offset palety barw zaczyna się od 0x010058 (65624)
- wykorzystano tylko 253 barwy (3036 bajtów).

ETAP DRUGI - Obrazek własny.

Wiedząc w jaki sposób i gdzie dane są zapisane mogę spróbować podmienić je na swoje.
Najpierw napisałem algorytm który realizuje kompresję RLE. Do testów posłużył mi oryginalny obrazek który wyciągnąłem z pliku BootImage. Po jego napisaniu i przetestowaniu stwierdziłem że mój algorytm okazał się wydajniejszy i plik był o kilkanaście bajtów krótszy. Zaciekawiło mnie to. Jednak dekompresor który napisałem wcześniej odtwarzał obrazek bezbłędnie. Dodatkowo upewniłem że wszystko jest w porządku poprzez doklejenie moich danych do pliku BootImage gdzie brakujące bajty zastąpiłem wartością 0x80 (dekompresor napotykając takie dane ignoruje je). Obrazek przy starcie systemu OS4.1 pokazał się bez najmniejszych problemów a więc pełny sukces. Jednak nie do końca. Długość ta miała znaczenie ale o tym później.
Gdy stwierdziłem że wszystko działa jak należy, zacząłem poszukiwania na sieci jakiegoś obrazka którym mógłbym zastąpić ten oryginalny. Padło na:

Ponieważ jestem fanem Mangi/Anime ten właśnie obrazek jakoś bardzo mi się spodobał. Wczytałem go do PPainta. Ten przekonwertował mi go do 8 bit, dodatkowo przekolorowałem do 253 barw, oraz odpowiednio wykadrowałem. Zapisałem jako BMP. Następnie taki plik pociąłem oddzielając grafikę od palety. Dane obrazka skompresowałem, paletę przekonwertowałem. Następnie wszystko wkleiłem gdzie trzeba do pliku BootImage sprawdzając czy nigdzie się nie pomyliłem. Ponieważ plik po kompresji był znacznie krótszy dokleiłem odpowiednią ilość 0x80 jak poprzednio. Przegrałem plik gdzie trzeba. Odpaliłem Sam440 i moim oczom ukazał się... No właśnie nie wiem, bo w żaden sposób nie przedstawiało to tego czego oczekiwałem. Jednym słowem wojna mrówek bijąca się kaszkę mannę. Co się stało? Co poszło nie tak? Gdzie się pomyliłem? Długo się nad tym zastanawiałem sprawdzając po trzykroć algorytmy oraz umiejscowienie danych. Nic nie pomagało. Po pewnym czasie przypomniałem sobie o tej różnicy długości pliku testując mój kompresor i nie dawało mi to spokoju. Skądś ta różnica musiała wynikać. Po długich analizach

dostałem olśnienia. Mój algorytm kompresował dane całościowo. I tu właśnie był błąd. Oryginalny plik każdą linię obrazu miał kompresowaną oddzielnie. Aby sprawdzić moją teorię nie przerabiałem algorytmu kompresora tylko dodałem do mojego obrazka szary gradient w postaci pionowych pasów.

Był on tak dobrany aby mieć pewność że kompresji zostanie poddana następna linia. Dlaczego to nie wyszło wcześniej? Oryginalny obrazek posiada napis nie na jednolitym tle tylko gradiencie kołowym. To on sprawił że dane kompresowały się w odpowiedni sposób. Po dodaniu gradientu w końcu mogłem się cieszyć własnym obrazkiem przy starcie systemu AmigaOS4.1.

Dla dalszych testów wykonałem kilka testów z różnymi obrazkami z czego dwa umieściłem na OS4 Depot.

Myślałem nad tym czy nie napisać programu z poprawionym algorytmem kompresji i konwerterem palety barw który wykonywałby plik Bootimage automatycznie z dowolnego obrazka. Jednak w trakcie prac wyszedł jeszcze jeden problem. Własny obrazek po kompresji musi zawierać się maksymalnie w takiej ilości bajtów jak obrazek oryginalny. Często przy przygotowywaniu innych obrazków okazywało się że po kompresji plik był dłuższy od zakładanego limitu. Im był bardziej szczegółowy tym bardziej działało to na niekorzyść. Należało go więc sztucznie modyfikować przez usuwanie pewnych fragmentów lub zmniejszanie ilości barw aż otrzyma się zadowalający wynik. Co prawie zawsze skutkowało tym, że obrazek po takich zabiegach nie wyglądał już jak bym chciał.

AmigaOS 4.1

Myślę że każdy kto zna się choć trochę na programowaniu na podstawie informacji które tutaj przedstawiłem i chciałby taki program napisać zrobi to bez trudu. W razie czego mogę służyć pomocą.

Aleph One / Marathon

truman burbank

Firmę Bungie zapewne część z was kojarzy z takimi konsolowymi hitami jak “Halo” czy “Destiny”. Dla mnie Bungie to przede wszystkim “Myth”, jeszcze z czasów przed przejęciem firmy przez Microsoft, oraz właśnie “Marathon”.

W “jedynkę” (”Marathon” - 1994 r.) nie grałem - była tylko na komputery Macintosh. Grałem jedynie w demo drugiej części (”Marathon 2: Durandal” - 1995 r.). Wrażenie jakie na mnie zrobiła w tym czasie? Żadne, przeciętna strzelanka FPP, choć opinie "makówkarzy" z pewnością są inne. Dla nich to wtedy była gra kultowa - mogło tak być, bo gier na komputery Apple zawsze było mniej. Trzecią część cyklu (”Marathon Infinity” - 1996 r.) nawet nie odnotowałem w świadomości, wyszła tylko dla Apple Macintosh. Dla mnie wtedy na topie byli następcy “Doom 2”: “Duke Nukem 3D” (1996) , czy powstałe na jego engine (Build) takie hity jak “Shadow Warrior” i “Blood” (1997).

Aleph One to silnik do gier, który powstał po tym jak firma Bungie opublikowała źródła do gry Marathon 2. Projekt jest w miarę aktywny, a produkt dopracowany.
Ma własny portal, pod adresem: https://alephone.lhowon.org

W górnej części głównej strony dostępna jest trylogia Marathon w wersji dla MacOS X, Windows i Linuksa. Są to gotowce do ograniа (gra plus silnik Aleph One). Niżej, w sekcji “More games”, podlinkowane są scenariusze oraz silnik Aleph One. Scenariusze to kompletne dane do pogrania w trylogię “Marathon” ale także gry stworzone przez społeczność. Czyli stąd bierzemy “pełniaki”, grzebanie po innych miejscach w internecie jest stratą czasu. Natomiast silnik jest w wersji dla MacOS X, Windows i Linuksa, więc amigowego musimy szukać gdzie indziej.

Jeżeli chodzi o wersję amigową - to nie jest tak że Marathon jest dostępny dla nas od niedawna. Mieliśmy go już w 2005 roku, ale okazja do odświeżenia sobie gry jest dobra. Capehill zrobił nową wersję, polecam odwiedzić jego stronę z projektami: http://capehill.kapsi.fi
Tam w sekcji Ports należy pobrać pozycje spod linków Aleph One (SDL2/MiniGL) i SDL2. Czyli silnik gry i biblioteki dla systemu. (prawdopodobnie można je będzie znaleźć także na OS4 Depot).
Skąd wziąć gry - podałem wyżej.

Instalacja gry to zainstalowanie w systemie SDL2 (oczywiście wersja dla użytkownika, nie SDK) oraz rozpakowanie gry.
Następnie do katalogu z grą trzeba wrzucić Aleph One (wystarczy sam plik wykonywalny). I to jego uruchamiamy (szuka gry w katalogu w którym jest).

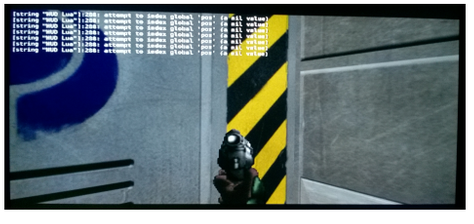
Fabuła gry to odległa przyszłość. Kosmos, Obcy, walka o przetrwanie. Standard. Gry jednak mają swoje uniwersum (Traxus) i ma to swoje odzwierciedlenie w rozgrywce.
Wspomniana fabuła nie jest tylko jak w innych FPP umocowaniem mającym być jedynie pretekstem do eksterminacji wszystkiego co się rusza. Tu mamy faktycznie wątki.
Podejrzewam że jednak większość nie ma zamiaru aż tak się wczuwać w grę i śledzić niuanse - dla chętnych zarys fabuły jest tutaj:
http://marathongame.wikia.com/wiki/Marathon_Trilogy

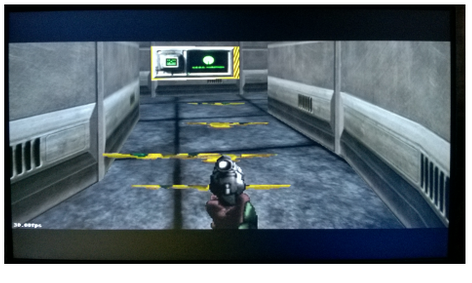
Skupiam się cały czas na “jedynce” żeby była jasność, bo od pierwszej części polecam zacząć. Fabularnie w niej jesteśmy oficerem ochrony na statku i bronimy go przed abordażem Obcych (niewolnicza rasa Pfhor).

Po uruchomieniu gry trzeba ją skonfigurować.
Polecam zmniejszenie czułości myszki (ja zmniejszyłem przez pół). Także włączenie celownika bardzo ułatwia rozgrywkę.
Można sobie przedefiniować klawisze - jest wiele opinii że są niestandardowo obłożone. Nie zgadzam się z tym. Może niektóre nie są intuicyjne (np. klawisz Tab do interakcji z przedmiotami) ale dla mnie rozłożenie jest wygodne.
Poruszanie - klasyczne WSAD.
LPM i spacja - strzelanie
F,R - zmiana broni

F2 i F3 - zmiana rozdzielczości gry w czasie gry (nie polecam)
F6 - przełączanie widoku FPP/TPP
F7 - zoom
F8 - włącz./wyłącz. celownika
F11 i F12 - zmiana jasności
Enter - kasowanie konsoli jeśli przypadkowo włączyliśmy ją
,. - regulacja głośności dźwięku
(<> - łatwiej zapamiętać wzrokowo)

Komunikaty u góry ekranu to jeden z kilku błędów w grze. Jak widać na ostatnim zrzucie udało się uruchomić raz bez błędu, niemniej błąd jest ewidentny i powtarzalny. Napisy przeszkadzają w rozgrywce.



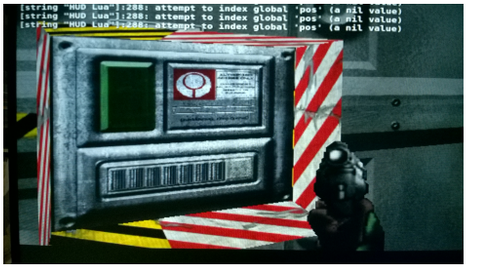


Na powyższych zrzutach graficznie wygląda to dużo lepiej niż oryginał z lat 90-tych. Obrazki przestawiają grę w trybach Software (pierwszy) oraz OpenGL. Na tę chwilę dostępny jest tryb OpenGL (classic), ustawienie w wersji (shader) nie zadziała.
Tryb Software (pikseloza) jest wolny i nawet nie ma co się nim zaszczepiać. Natomiast wsparcie nowych kart graficznych wnosi bardzo wiele - zarówno płynność rozgrywki, jak i oprawa są bardzo dobre. Trzeba patrzeć na trylogię “Marathon” nie jak na gry z lat 90-tych, ponieważ są to gry odnowione. Wygładzone tekstury, wysokie rozdzielczości, mnogość opcji konfiguracyjnych (można ustawić nawet kolor celownika broni) - to robi swoje, i z pewnością można będzie do “Marathona” w przyszłości wracać.

Na sam początek jest drugi feler wersji amigowej - miałem zamiast normalnych przeciwników jakieś pluskwy na podłodze, które ciężko trafić. Dostępna pukawka o długim czasie przeładowania i jeszcze dłuższym zmiany magazynku utrudnia sprawę.
Dalej jest już “z górki”. Przeciwnicy są drętwi i nieco pokraczni, wyglądają z włóczniami na łatwy cel. I po części tak jest, co nie znaczy że są niegroźni - kilka uderzeń jest śmiertelnych, a przeciwnik atakuje grupami.

Byłem bardzo zaskoczony gdy podczas kolejnej rozgrywki nie widziałem wrogów. Doszedłem do tego że “pluskwy” to szczątkowe postaci przeciwników. Żeby mieć ich normalnych (także na samym początku gry, nie jako “pluskwy”) trzeba zmienić ustawienie grafiki. Drogą prób doszedłem do tego że musi być wyłączony Z-Buffer (ustawienie znajduje się w Graphics -> Rendering Options -> Advanced). To jest istotna wskazówka, bez tego ukończenie gry nie jest możliwe. Tylko wtedy nie tylko nie będziemy mieć problemu z widocznością przeciwników, ale prawidłowo zadziała widok TPP (zza pleców).
Kolejne niedoskonałości amigowej wersji to nie działający celownik broni. Nie działa też prawidłowo wyświetlanie terminali (zawartość jest pomniejszona i rozmazana). Nie działają również ustawienia dla zmiany jasności. To akurat jest drobiazg, ale dużą wadą jest brak interfejsu bohatera (dół ekranu - radar i broń). Bardzo możliwe że te błędy są bardzo łatwe do usunięcia, i to się stanie.

Można spotkać trzy rodzaje paneli, podaję je, podobnie jak wcześniej podstawy klawiszologii, by już na początku nie odbić się i nie zniechęcić.



Pierwszy zawiera wskazówki i mapy, drugi to włącznik, trzeci to zapis gry. Zapis odbywa się bez potwierdzenia. Udogodnieniem jest to, że przy wczytywaniu na liście zapisów są (w formie fragmentów mapy) miniaturki lokacji w których byliśmy w danym momencie.
Marathon nie posiada kodów, nie ma też opcji skakanie - co nie znaczy że

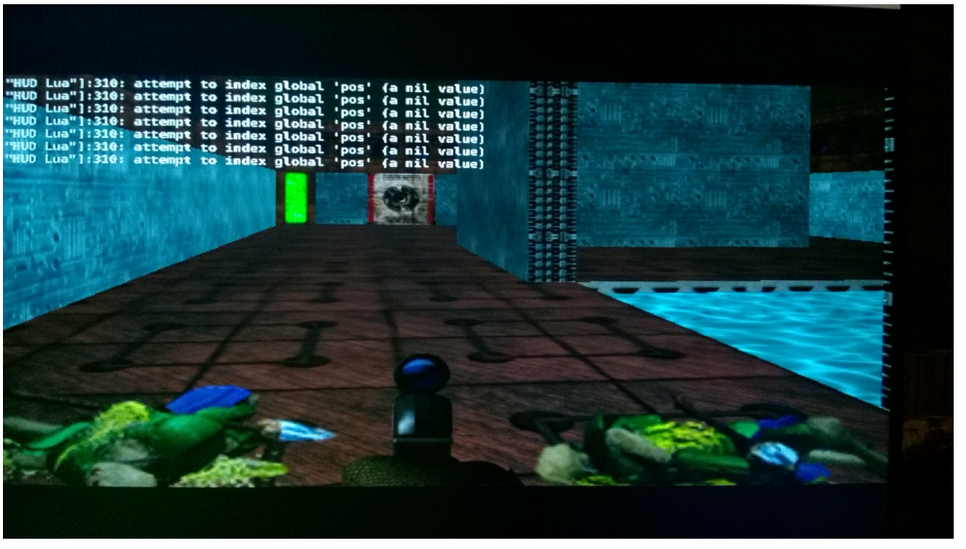
gra nie ma swoich ukrytych tajemnic. Można o nich poczytać na “Marathon Spolier Guide” pod adresem: http://marathon.bungie.org/spoiler/general.shtml

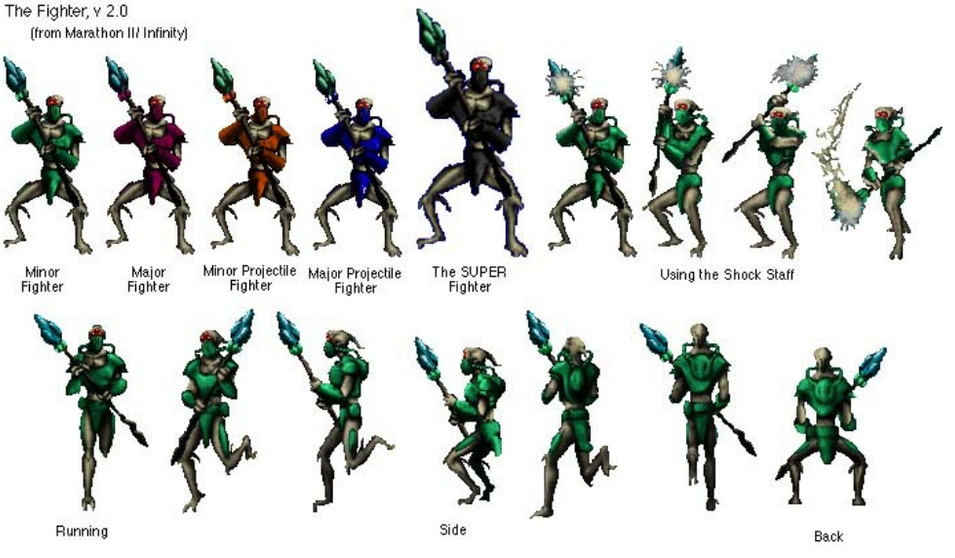
Rozgrywkę oceniam jako trudną. Kojące dźwięki, przywodzące momentami na myśl takie tytuły jak “Privateer”, po pewnym czasie stały się dla mnie klaustrofobiczne - straciłem dużo czasu na odkrycie “jak pójść dalej”. Po otwarciu ukrytego przejścia eksploracja wąskich korytarzyków z systemem wind i zabijającym mnie skurczybykiem w czerwonym płaszczu, oraz konieczność powtarzania tego do skutku - okazały się frustrujące. A wychodzić na kolejne poziomy trzeba, bo przeciwnicy odradzają się. Przechodzę tę grę po trochu, ale nie wiem czy wytrwam.

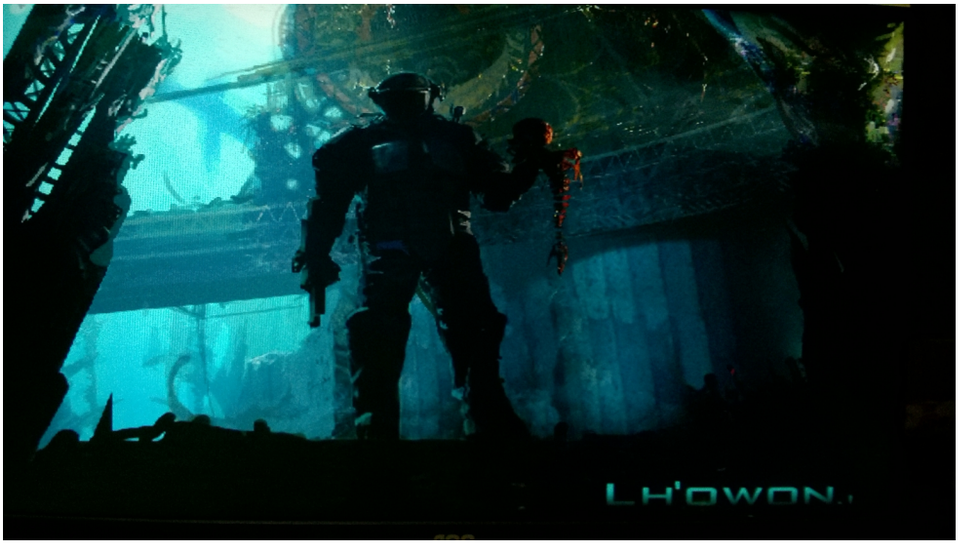
O wiele bardziej zachęcająco prezentuje się Marathon 2.
“Dwójka” jest lepsza o tyle, że wreszcie są otwarte przestrzenie. Jest trochę inny klimat, przynajmniej ja czułem się silniejszy. Niemniej sytuacja ta sama co w pierwszej odsłonie gry - ten port Aleph One po prostu wymaga poprawki.
Części trzeciej nie uruchamiałem ale z pewnością działa, to jest ten sam silnik.

Jakie wnioski?
“Marathon” mnie nie porwał. Może gdybym ograł go “w swoim czasie” miałbym inne podejście.
Jak najbardziej Aleph One jest grywalny i atrakcyjny współcześnie, niestety niedoróbki portu psują odbiór rozgrywki. Nie są to błędy wykluczające ukończenie gier, niemniej po uruchomieniu wersji pod Windows i porównaniu moje oczekiwania są większe.
Czekam na wersję poprawkową.

Ilustracje poniżej to kolejno: pierwszy poziom Marathon 2, przykładowi przeciwnicy (ilustracja z Wiki fanów) oraz klimatyczna plansza tytułowa “dwójki”







Hollywood

#1 Tagać, tagać, tagać…

ferin

Po przybyciu na party-plejs wygenerowaliśmy kilka fraktali na trawniku, skopaliśmy śmietnik i naubliżaliśmy staruszce...
Uh-oh, to nie ten rok, nie to miejsce i nie ten artykuł, ale jak już zacząłem i najwyraźniej ktoś to czyta, to zabiorę was na wyboistą jazdę Kniejami Ostrokrzewu, z angielska zwanego Hollywoodem.

Od razu zaznaczam, że jeśli ktoś oczekuje fulprofeski--super-hiper tutoriala to może przestać czytać w tym miejscu i udać się pod ten adres: forums.hollywood--mal.com, tam znajdzie (albo i nie) coś odpowiedniego dla siebie.

A wracając na party-plejs... hmm i znowu. Zacznijmy jeszcze raz.
A wracając do rzeczy istotnych, będzie to bardziej opis moich doświadczeń podczas prób nauczenia się tego języka programowania, niż tutorial. Taki tekstowy odpowiednik filmów na jutjubie, które ludziki nagrywają podczas swoich pierwszych, a czasami ostatnich razów, na przykład z rozbrajania niewybuchów czy denerwowania swoich żon (nie wiem co gorsze).

Ostrzegam i zastrzegam, że nie mam żadnego planu, będzie to pełen spontan, z tego powodu nawet ja nie wiem o czym, kiedy i czy w ogóle będzie następna część. Wiecie jak to jest - obiecanki cacanki a czekającemu smutek.
Dla równowagi, jeżeli już coś napiszę będę się starał z całych sił żeby było jasno, czytelnie, zrozumiale i żeby można coś z nabytą w znoju wiedzą zrobić. Czyli taki praktyczny poradnik raczej niż sucha wiedza, z którą nie bardzo wiadomo co zrobić - takie zresztą mam odczucie podczas czytania manuala do Hollywood. Nie zrozumcie mnie źle, manual jest wporzo, aczkolwiek pisany przez autora i czasami rzeczy dla niego oczywiste, które w nim pomija, dla mnie - nieprogramisty - są niezłą zagwozdką.

Z rzeczy organizacyjnych muszę jeszcze dodać, że przeprowadzenie kompotów było... Do stu zielonych Superfrogów! Co się ze mną dzieje?! ...Muszę jeszcze dodać, że będę się starał pisać tak żeby każdy użytkownik amigowego en-gje, bez względu na smak/obóz (jak zwał tak zwał), mógł bez jakichkolwiek przeróbek, ściągania czcionek itp. uruchomić/skompilować taki skrypt u siebie.

Przechodząc do rzeczy napiszę szybko o podstawach, znajomość których ułatwi nam naukę Hollywood.

Kod źródłowy to nic innego jak zwykły plik tekstowy najlepiej w UTF-8, pisane w jakimkolwiek nadającym się edytorze tekstu.
Niestety GUI Hollywoodowe na Amidze nie posiada, tak jak wersja (tfu) pecetowa, wbudowanego edytora z podświetlaniem składni. Można sobie poradzić zakupując CubicIDE albo zasysając odpowiedni plik z sieci dla edytora Annotate.
Póki co nasz kod (ależ to brzmi ;)) będzie tak mało skomplikowany, że będę używał systemowego NotePada.

To jedziemy z tym koksem.
Pliki Hollywoodowe mają cztery rozszerzenia:
.hws - pliki źródłowe/skrypty
.hwa - applety
.hwd - pliki tworzone przez Designera
.hwp - pliki wtyczek/pluginów

Oczywiście są jeszcze pliki wykonywalne ale one nie mają żadnego konkretnego rozszerzenia, zależy ono od systemu dla jakiego został skompilowany nasz program.

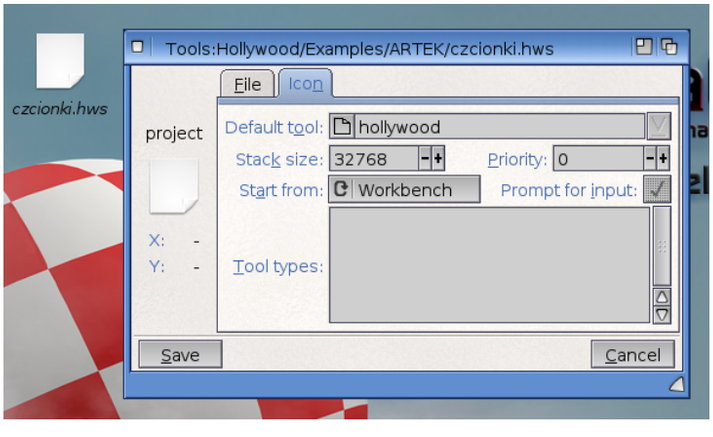
Uruchomienie naszego skryptu możliwe jest na co najmniej dwa sposoby: poprzez przycisk "Display" w GUI, albo poprzez dwuklik na plik źródłowy.
W drugim przypadku trzeba zmienić "Domyślne narzędzie/Default tool" w ikonce pliku ze skryptem z "SYS:Utilities/MultiView" na "Hollywood".

Mi najbardziej odpowiada ostatnia możliwość ale wy zrobicie jak wam będzie najwygodniej. Kompilacji natomiast możemy dokonać poprzez klik w "Compile" w GUI Hollywooda albo z linii poleceń.
Nie będę opisywał opcji dostępnych dla kompilacji bo dostępne są w sieci pod adresem: http://www.hollywood--mal.com/docs/html/hollywood/ManualUsage.html.
Dodam tylko, że wszystkie te argumenty można zastosować do już skompilowanego programu poprzez dodanie odpowiednich tooltype'ów do ikony naszego programu albo przez dodanie ich w wierszu poleceń podczas uruchamiania z shella.

Odnośnie opcji kompilowania z wiersza poleceń napiszę tylko, że główny program/interpreter Hollywood to nic więcej jak komenda, których wiele w systemowym katalogu C:.
Nadmienię tym co nie czytają manuali, że bezwzględnie nie można tym plikiem dzielić się z nikim, to jest serce Hollywood chronione prawami autorskimi.

Dobra, dość kwasu, czas zacząć kodowanie.

Tworzymy plik tekstowy.
Ja nazwałem swój czcionki.hws, wywaliłem na blat WB i zmieniłem tooltype na Hollywood, dla szybkości i wygody dostępu:



Niestety muszę jeszcze posmęcić, bo jest kilka rzeczy o których musimy wiedzieć pisząc skrypty dla Hollywood.

Na początku każdego skryptu musi znajdować się komenda preprocesora **@VERSION**.
Mówi ona interpreterowi jaka wersja Hollywood jest konieczna do skompilowania naszego skryptu. Bierze się to stąd, że różne możliwości dodawane były w różnych wersjach Hollywood i na nas spada odpowiedzialność aby dokładnie czytać dokumentację i umieścić tam odpowiedni numerek.

Każda opisywana funkcja ma podaną wersję wymaganego Hollywood, a nawet poszczególne argumenty przekazywane do funkcji pojawiają się w różnych wersjach Hollywood.
Dla przykładu funkcja Font wymaga wersji 4.5:

NAME

FONT -- preload a font for later use (V4.5)

SYNOPSIS

```
@FONT id, fontname$, size[, table]
```

Gwoli jasności, ta komenda nie musi być w pierwszym wierszu pliku ze skryptem, bo tam może być komentarz który chcieliśmy umieścić dla naszej lub innych wygody. Ale musi znajdować się na pierwszej interpretowanej linii skryptu. Mam nadzieję że czujecie różnicę.

Póki co biednie, ale nasz plik wygląda tak jak na zrzucie (Zrzut 3).

Komentarze

Przy tej okazji muszę napisać kilka słów o następnym elemencie pojawiającym się w naszym kodzie, a mowa tu o komentarzach. Jak widzicie po średniku jest wpis:
- podałem wersję 1,0 bo nie ma tu nic co wymagałoby wyższej.

W Hollywood występują dwa rodzaje komentarzy, wielo i jedno liniowe.
Jednoliniowy komentarz następuje po średniku ";" i może zająć przestrzeń tylko do końca linii, taki komentarz może znajdować się na samym początku linii albo tak jak w naszym przykładzie znajdować się po funkcji i jej argumentach.

Drugim rodzajem komentarzy jest ten wieloliniowy i zaczyna się od sekwencji znaków "/*" a kończy się na "*/". Wszystko pomiędzy nimi będzie ignorowane przez Hollywood, więc możemy tam wpisać co nam się żywnie podoba, ale najlepiej coś co pomoże nam odnaleźć się w naszym kodzie. Ilość linii nie jest ograniczona i może zająć jeden albo nieskończenie wiele wierszy.

```
@VERSION 1,0                                    ; podałem wersję 1,0 bo nie ma tu nic co wymagałoby wyższej
```

Zrzut 3

```
@VERSION 1,0                                    ; podałem wersję 1,0 bo nie ma tu nic co wymagałoby wyższej
@DISPLAY {width=800, height=600, borderless = True}    ; szerokość 800 pikseli, wysokość 600, okno bez ramki
```

Zrzut 4

```
@VERSION 4,7                                    ; zmiana wersji na 4,7
@DISPLAY {width=800, height=600, borderless = True}    ; szerokość 800 pikseli, wysokość 600, okno bez ramki
@FONT 1, "Arial", 50, {Engine=#FONTENGINE_INBUILT}     /* Ładujemy czcionkę Arial o rozmiarze 50 i określamy sposób  *
                                                        * wyświetlania na wbudowany w Hollywood:                    *
                                                        * Engine=#FONTENGINE_INBUILT.                               *
                                                        * a nasza czcionka będzie dostępna pod identyfikatorem 1    */
@FONT 2, "Arial", 20, {Engine=#FONTENGINE_INBUILT}     ; Jak wyżej tylko rozmiar 20 i identyfikator 2
```

Zrzut 5

@DISPLAY *[id,] table*

Jak już przebrnęliśmy przez absolutną podstawę, to musimy teraz określić wielkość i rodzaj okna w jakim będzie wyświetlany nasz program.(Zrzut 4)
Służy do tego kolejna funkcja preprocesora @DISPLAY. Jeszcze jedna kwestia, autor Hollywood zaleca aby komendy preprocesora zapisywać dużymi literami. Wszystkie funkcje preprocesora zaczynają się małpą "@".

Komenda @DISPLAY służy do określenia rodzaju, rozmiaru, tła i chyba z setki innych opcji dla naszego okna albo ekranu naszego programu.
Po cały opis wysyłam pod: http://www.hollywood--mal.com/docs/html/hollywood/atDISPLAY.html.

Ta komenda jest dobrym przykładem pokazującym jak poszczególne argumenty pojawiały się w różnych wersjach Hollywood.
Nasze okienko będzie miało szerokość 800 i wysokość 600 pikseli i nie będzie miało ramki. Dodajmy teraz jakąś czcionkę, którą będziemy mogli pisać w naszym okienku. Na samym początku zwróćcie uwagę, że zmienił się numer wersji wymaganego Hollywood, powodem jest użycie "Engine=#FONTENGINE_INBUILT" który to argument pojawił się w Hollywood 4.7.

@FONT *id, fontname$, size[, table]*

Teraz rozpracujmy ten rozkaz (Zrzut 5).
Komenda ta ładuje nam font o podanej nazwie i rozmiarze i przypisuje mu identyfikator. W naszym wypadku są dwie takie komendy. Pierwsza przypisuje identyfikatorowi "1" czcionkę Arial o wielkości 50 pikseli, druga zaś identyfikatorowi "2" tę samą czcionkę ale o wielkości 20 pikseli.
W obu przypadkach na końcu występuje w nawiasie klamrowym atrybut funkcji:
"Engine=#FONTENGINE_INBUILT"
Powoduje on że wykorzystywany jest wbudowany w Hollywood sposób wyświetlania czcionek, co gwarantuje że tekst będzie wyglądał tak samo na każdym systemie dla jakiego skompilujemy nasz program.
Ta komenda ma tę właściwość, że automatycznie dołącza czcionki do naszego programu. Daje nam to pewność, że nie zabraknie czcionki wykorzystanej w naszym programie. Z drugiej strony musimy uważać jakie czcionki używamy i jakim prawom autorskim podlegają. Ponad to ma to tę zaletę że czcionki są dostępne od samego początku wykonywania się programu, nie trzeba ich "ładować".

W Hollywood poza używaniem czcionek bitmapowych i trutajpów zassanych z sieci mamy jeszcze jedną możliwość, a mianowicie możemy użyć czcionki wbudowanej. Ma to tę zaletę, że nie musimy linkować plików z czcionkami, ani ładować ich z dysku użytkownika, ta czcionka jest wszędzie i zawsze dostępna. Dodatkowym atutem jest gwarancja tego samego wyglądu na wszelkich systemach pod jakimi może działać nasz program.
Ograniczeniem natomiast jest to, że mamy tylko po jednym kroju czcionek bitmapowych i truetajpów do wyboru i żadnych znaków narodowych, czyli możemy póki co zapomnieć o "ę", "ó" i tak dalej. Może się to kiedyś zmieni.
Jeżeli wspomniane ograniczenia nie będą mieć wpływu na nasz program, to myślę, że jest to całkiem dobry wybór. Bo bardzo łatwy w użyciu i nie nastręcza kłopotów z ładowaniem plików z dysku (co może być dość błędogenne w przypadku kompilowania pod wiele systemów, ale o tym napiszę nieco później), ani zmartwień o licencje w przypadku ich linkowania.

Żeby zadeklarować chęć użycia wbudowanej w Hollywood czcionki zamiast nazwy kroju używamy znacznika "#" plus nazwa z dużej litery.
Mamy do wyboru pięć, a właściwie póki co cztery rodzaje czcionek:
#SANS - otwiera wbudowaną czcionkę trutype bezszeryfową
#SERIF - otwiera wbudowaną czcionkę trutype z szeryfami
#MONOSPACE - otwiera wbudowaną czcionkę trutype o stałej szerokości znaku
#TRUETYPE_DEFAULT - póki co daje taki sam wynik jak użycie #SANS (stąd cztery a nie pięć rodzajów)
#BITMAP_DEFAULT - otwiera wbudowaną czcionkę bitmapową o rozmiarze 8 pikseli, i w żaden sposób na razie nie możemy tego zmienić.

Zmieńmy teraz czcionkę Arial 20 pikseli na wbudowaną w program. Dodatkowo żeby sprawdzić jak wygląda czcionka bitmapowa dodamy sobie jeden wiersz z poleceniem jej załadowania. I tak nasza źródłówka wygląda tak jak na zrzucie (Zrzut 6).
Jak widzicie użycie jest bardzo proste, z jedną uwagą. Nawet jeśli mamy do wyboru tylko czcionkę bitmapową o rozmiarze 8 pikseli, to i tak musimy podać rozmiar funkcji @FONT, bo przy kompilacji wyskoczy nam błąd. Poza tym w przypadku czcionki wbudowanej nie musimy podawać atrybutu "Engine=#FONTENGINE_INBUILT" bo jest on wybierany automatycznie.

OpenFont *(id, fontname$, size[, table])*

Żeby umożliwić użycie jakiegoś kroju czcionek w naszym programie poza funkcją preprocesora @FONT, możemy użyć funkcji OpenFont.
Jedyna różnica po między tymi rozkazami jest taka, że czcionki załadowane przez OpenFont są dostępne od miejsca użycia tej funkcji.
Czyli czcionki nie będą dostępne od samego początku działania programu, a dopiero gdy nastąpi wykonanie rozkazu OpenFont.
Ponadto, jak wspomniałem wyżej. funkcja "@FONT" dołącza/linkuje czcionki do pliku wykonywalnego naszego programu. To zachowanie można zmienić ale musimy o tym pamiętać, bo zmienia to wielkość pliku wykonywalnego, szczególnie w wypadku dołączenia wielu krojów, i nakłada na nas odpowiedzialność stosowania się do licencji.
Oczywiście zdaję sobie sprawę że tworząc nasze programiki nikt takimi bzdurami jak prawa autorskie nie będzie się przejmował, ale mając przed oczami świetlaną przyszłość naszych zaawansowanych wieloplatformowych tworów, muszę o tym wspomnieć żeby uchronić nas przed hordą wygłodniałych prawników w chwili gdy nasz soft odniesie komercyjny sukces ;).

```
@VERSION 4,7                                    ; zmiana wersji na 4,7
@DISPLAY {width=800, height=600, borderless = True}  ; szerokość 800 pikseli, wysokość 600, okno bez ramki
@FONT 1, "Arial", 50, {Engine=#FONTENGINE_INBUILT}   /* Ładujemy czcionkę Arial o rozmiarze 50 i określamy sposób  *
                                                     * wyświetlania na wbudowany w Hollywood:                    *
                                                     * Engine=#FONTENGINE_INBUILT.                               *
                                                     * a nasza czcionka będzie dostępna pod identyfikatorem 1    */
@FONT 2, #SERIF, 20                             ; Czcionka szeryfowa wbudowana w Hollywood o rozmiarze 20
@FONT 3, #BITMAP_DEFAULT, 8                     ; Czcionka bitmapowa wbudowana w Hollywood o rozmiarze 8
```

Zrzut 6

```
@VERSION 4,7                                        ; zmiana wersji na 4,7
@DISPLAY {width=800, height=600, borderless = True} ; szerokość 800 pikseli, wysokość 600, okno bez ramki
@FONT 1, "Arial", 50, {Engine=#FONTENGINE_INBUILT}  /* Ładujemy czcionkę Arial o rozmiarze 50 i określamy sposób  *
                                                    * wyświetlania na wbudowany w Hollywood:                    *
                                                    * Engine=#FONTENGINE_INBUILT.                               *
                                                    * a nasza czcionka będzie dostępna pod identyfikatorem 1    */
@FONT 2, #SERIF, 20                                 ; Czcionka szeryfowa wbudowana w Hollywood o rozmiarze 20
@FONT 3, #BITMAP_DEFAULT, 8                         ; Czcionka bitmapowa wbudowana w Hollywood o rozmiarze 8

UseFont(1)                                          ;użyj czcionki o identyfikatorze 1 - w tym wypadku Arial 50 pikseli
```

Zrzut 7

Jak dotąd udało nam się otworzyć okno naszego programu i udostępnić czcionki, które będziemy mogli używać w naszym programie.
Teraz zajmiemy się funkcjami które będą przypisywać krój i w zależności od potrzeb zmieniać jego właściwości (Zrzut 7).

UseFont *(id)*

Rozkaz ten zmienia czcionkę domyślną na tę określoną identyfikatorem w nawiasach, w naszym przykładzie jest to 1, czyli Arial o rozmiarze 50 pikseli. Styl czcionki zostanie przywrócony do domyślnego po użyciu tej funkcji. Na przykład jeśli zmieniliśmy krój naszej czcionki na pogrubioną kursywę to po zastosowaniu tej funkcji zostanie on zmieniony na zwykły niepogrubiony. Ta funkcja jest bardzo prosta ale i ograniczona, bo narzuca nam krój, rozmiar itd. bez możliwości jakichkolwiek zmian w jej obrębie. Czyli w naszym przykładzie jeśli użyjemy UseFont(1) to nasz tekst będzie pisany Arialem o wysokości 50 pikseli i bez użycia innych rozkazów w żaden sposób tego nie zmienimy.

SetFont *(font$, size[, table])*

Aby móc zmienić różne rodzaje czcionki, czy ją samą na inną, użyjemy rozkazu SetFont. Atrybut font$ określa czcionkę jaką chcemy użyć, tu muszę wrócić do "błędogenności" wspomnianej przy opisywaniu komendy @FONT. Aby móc załadować konkretny krój czcionki z dysku trzeba trzymać się pewnych reguł narzuconych przez Hollywood. Podając nazwę czcionki wpisujemy jej nazwę, a nie ścieżkę dostępu (choć jest wyjątek od tej reguły). Nazwa musi być identyczna z nazwą pliku na dysku bez rozszerzenia. I tak jeżeli chcemy użyć czcionki Arial bold italic, to jako atrybut musimy wpisać "Arial Bold Italic", zwróćcie uwagę że nazwa czcionki i jej atrybuty muszą być oddzielone spacją, a całość zamknięta w cudzysłowie. Wyjątek stanowi przypadek, kiedy chcemy użyć czcionki wektorowej TTF, która znajduje się na dysku wraz z wbudowanym w Hollywood sposobem wyświetlania czcionek {ENGINE=#FONTENGINE_INBUILT}, wtedy to możemy podać całą ścieżkę dostępu wraz z nazwą pliku czcionki z rozszerzeniem. To tyczy się jedynie czcionek TTF.
Dla przykładu, możemy wtedy podać jako argument funkcji SetFont("SYS:Fonts/_ttf/Arial.TTF", 30, {Engine=#FONTENGINE_INBUILT}).
Błędogenność wynika stąd, że różne systemy instalują czcionki pod różnymi nazwami. Dla przykładu gdy plik czcionki ma nazwę arialbolditalic to jesteśmy pozamiatani, bo nie dosyć że nie ma spacji to cechy bold i italic są pisane z małych liter (Wielkość liter w nazwach pod niektórymi systemami też ma znaczenie).
Więcej na temat jak radzić sobie z przyjazną Hollywoodowi instalacją i użyciem czcionek znajdziecie pod poniższymi adresami:

http://www.hollywood-mal.com/docs/html/hollywood/PrgG2KFonts.html

http://www.hollywood-mal.com/docs/html/hollywood/FontSpecification.html

Print/NPrint *(var, ...)*

Zastosujemy teraz w naszym programie rozkaz wyświetlający tekst na ekranie, abyśmy mogli w końcu zobaczyć wynik naszej pracy w działaniu.
Znacznie lepiej się wtedy pracuje czy wprowadza poprawki no i usuwa błędy. Do tego celu użyjemy funkcji Print i NPrint. Opisuję obydwie jednocześnie bo są prawie identyczne. Jedyna różnica jest taka, że NPrint dodaje znak końca linii.

Za pomocą tej funkcji możemy wyświetlić zawartość atrybutu "var", cokolwiek to jest - ciąg znaków, liczby, tablice czy wartość zmiennych, które to mogą być całymi plikami ładowanymi do odczytania. Rozkaz ten zaczyna wyświetlać tekst w obecnym położeniu kursora, czyli w naszym wypadku lewy górny róg. Można to zmienić ale o tym kiedy indziej. Funkcja potrafi interpretować kody ucieczki (ang. escape codes), jak i znaczniki formatujące tekst znane choćby z for internetowych, takie jak [b][/b], [i][/i], [u][/u], a poza tym:
[shadow=color,size,direction][/shadow] - cień,
[edge=color,size][/edge] - obrys krawędzi,
[color=color][/color] - kolor czcionki.
Trzy pierwsze nie wymagają chyba opisu, a pozostałe trzy opiszę dokładniej trochę później przy opisie następnych funkcji.

```
\a    Ring the system bell
\b    Back space
\f    Form feed
\n    Newline character
\r    Carriage return
\t    Horizontal tab
\v    Vertical tab
\\    Backslash
\"    Double quote
\'    Single quote
\?    Question mark
\[    Square bracket open
\]    Square bracket close
\xxx  Code point
```

“escape codes”

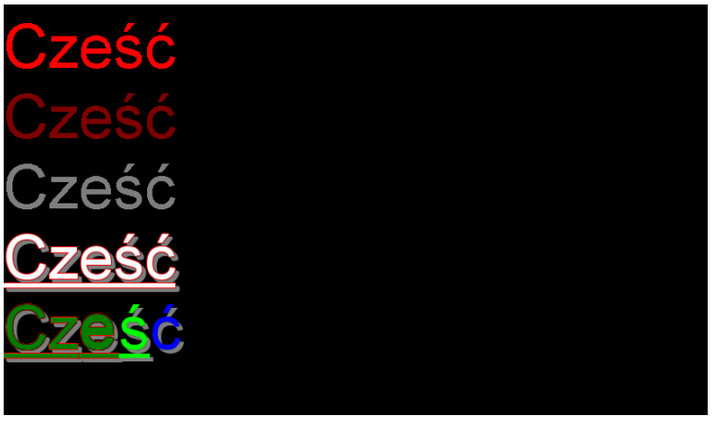
```
@VERSION 4,7                                          ; zmiana wersji na 4,7
@DISPLAY {width=800, height=600, borderless = True}   ; szerokość 800 pikseli, wysokość 600, okno bez ramki
@FONT 1, "Arial", 50, {Engine=#FONTENGINE_INBUILT}    /* Ładujemy czcionkę Arial o rozmiarze 50 i określamy sposób  *
                                                      * wyświetlania na wbudowany w Hollywood:                    *
                                                      * Engine=#FONTENGINE_INBUILT.                               *
                                                      * a nasza czcionka będzie dostępna pod identyfikatorem 1    */
@FONT 2, #SERIF, 20                                   ; Czcionka szeryfowa wbudowana w Hollywood o rozmiarze 20
@FONT 3, #BITMAP_DEFAULT, 8                           ; Czcionka bitmapowa wbudowana w Hollywood o rozmiarze 8

UseFont(1)                                            ;użyj czcionki o identyfikatorze 1 - w tym wypadku Arial 50 pikseli
NPrint("Cześć")                                       ;nasz pierwszy tekst wyświetlony na ekranie funkcją NPrint
Print("C[edge=#RED, 1]z[b]e[/b][/edge][u]ś[/u][color=#GREEN]ć[/color]")
                                                      /* przykład zmiany stylu tekstu "w biegu", jak widać tagi     *
                                                      * formatujące można zagnieżdżać                              */

WaitLeftMouse()
End()
```

Zrzut 8

Dodajmy do naszego skryptu dwie funkcje:
NPrint("Cześć")
Print("C[edge=#RED,1]z[b]e[/b][/edge][u]ś[/u][color=#GREEN]ć[/color]") (Zrzut 8).

Pierwsza linia wypisuje po prostu Cześć i przesuwa kursor do następnej linii, za to w drugiej się dopiero dzieje. "Ce" jest nie zmienione, "Zet" ma czerwoną obwódkę o szerokości jednego piksela, podobnie jak "E", ale jest ono dodatkowo pogrubione. To żeby pokazać, że tagi formatujące można zagnieżdżać. "Ś" jest natomiast podkreślone, a "Ć" zielone (Zrzut 9).

Zrzut 9



Powyżej nieco powiększony zrzut okna, dla lepszego pokazania fontów, taka próbka możliwości. Spróbujcie w ramach ćwiczenia dla siebie napisać samodzielnie odpowiedni kod żeby uzyskać to samo.

SetFont, OpenFont, UseFont i @FONT ustawiają tylko rodzinę i rozmiar czcionki, żeby zmienić kolor i przeźroczystość, dodać cień czy obrys, albo wygładzanie opiszę teraz dwa nowe rozkazy SetFontColor i SetFontStyle. Ustawiają one styl domyślny dla tekstu, czyli nie musimy przy każdej funkcji Print czy NPrint wpisywać oddzielnie jeśli chcemy żeby nasz tekst był na przykład czerwony z zieloną obwódką.

SetFontColor *(color)* (Zrzut 10)

Jak nazwa wskazuje, rozkaz ten ustawia kolor czcionki aktualnej na inny niż domyślny (na marginesie domyślne kolory to biały dla czcionki i czarny dla tła). Aktualnej to znaczy, wszystko co napiszemy poprzez N/Print czy inne funkcje wyprowadzające tekst na ekran będzie miało wybrany przez nas kolor. Musimy pamiętać, że styl czcionki jest resetowany na domyślny po każdym użyciu funkcji SetFont czy UseFont. Czyli znowu musimy użyć SetFontColor czy SetFontStyle jeśli chcemy zmienić wygląd czcionki.
Wybierając kolor w zależności od potrzeby mamy do wyboru z i bez przeźroczystości. Wartości możemy podawać w formie dziesiętnej lub szesnastkowej. Jeżeli zdecydujemy się na szesnastkową to bezpośrednio przed podanymi wartościami wpisujemy znak dolara **"$"** - **$RRGGBB**, **$AARRGGBB**, a w przypadku drugiego sposobu wpisujemy **RGB** albo **ARGB** - RBG(0, 255, 255), w obu wypadkach pierwszy to bez, a drugi to z przeźroczystością. Kolory możemy też podawać w formie "stałych wbudowanych", czyli na przykład chcemy czerwony z 50% przezroczystości ARGB(128, #RED), to samo osiągniemy wpisując ARGB(128, $FF0000) albo ($80FF0000). Jak widać można łączyć sposoby wprowadzania wartości, ale są trzy duże ale, po pierwsze takie mieszanki mogą sprawiać kłopot a po drugie funkcja ta **NIE** przyjmuje zapisu w pełni dziesiętnego ARGB(128, 255, 0,0) nie wiem czy to błąd czy zamierzone zachowanie. I po trzecie jeśli w wypadku zapisu szesnastkowego chcemy użyć ARGB przed nawiasem to w jego wnętrzu musimy oddzielnie podać wartości dla przeźroczystości i koloru, w taki sposób: ARGB($80, $FF0000), czyli wartości muszą być oddzielone przecinkiem i każda mieć swojego $ - dolara. Poza tym zauważcie że sam znacznik A/RGB można pominąć jedynie w wypadku użycia formy w pełni szesnastkowej, w każdym innym wypadku musi się pojawić przed nawiasami z podanymi wartościami.
Wszystkie stałe wbudowane w Hollywood, bez względu na rodzaj, zawsze zaczynają się znakiem # i są pisane z dużych liter, na przykład #RED, #WHITE, #BLUE. Dzięki temu widząc tak opisany ciąg znaków w skrypcie

```
@VERSION 4,7                                        ; zmiana wersji na 4,7
@DISPLAY {width=800, height=600, borderless = True} ; szerokość 800 pikseli, wysokość 600, okno bez ramki
@FONT 1, "Arial", 80, {Engine=#FONTENGINE_INBUILT}  /* Ładujemy czcionkę Arial o rozmiarze 50 i określamy sposób  *
                                                    * wyświetlania na wbudowany w Hollywood:                    *
                                                    * Engine=#FONTENGINE_INBUILT.                               *
                                                    * a nasza czcionka będzie dostępna pod identyfikatorem 1    */
@FONT 2, #SERIF, 20                                 ; Czcionka szeryfowa wbudowana w Hollywood o rozmiarze 20
@FONT 3, #BITMAP_DEFAULT, 8                         ; Czcionka bitmapowa wbudowana w Hollywood o rozmiarze 8

UseFont(1)                                          ;użyj czcionki o identyfikatorze 1 - w tym wypadku Arial 50 pikseli
NPrint("Cześć")                                     ;nasz pierwszy tekst wyświetlony na ekranie funkcją NPrint

SetFontColor($FF0000)                               ;ustawiamy w hexach kolor naszej czcionki na czerwony
NPrint("Cześć")                                     ; i piszemy nią "Cześć"
SetFontColor(ARGB($80, $FF0000))                    ; zmieniamy przeźroczystość koloru na 50%
NPrint("Cześć")                                     ; i sprawdzamy jak wygląda
SetFontColor(RGB (255, 0, 0))                       ; ustawiamy kolor na czerwony ale tym razem dziesiętnie
NPrint("Cześć")                                     ; żeby pokazać jak to robić innym sposobem
SetFontColor(ARGB(128, #RED))                       ; zmieniamy na czerwony, półprzeźroczysty przy pomocy
NPrint("Cześć")                                     ; stałej wbudowanej
SetFontColor(ARGB(128, $FFFFFF))                    ; Teraz powrót do białej ale przeźroczystej
NPrint("Cześć")
SetFontColor(#WHITE)                                ; a tu pozbywamy się przeźroczystości.
NPrint("Cześć")


WaitLeftMouse()
End()
```

Zrzut 10

możecie być pewni, że jest to stała wbudowana. Oczywiście jeśli autor skryptu stosował się do zaleceń odnośnie stylizacji kodu Hollywood. Stałych w wbudowanych jest znacznie więcej, chętnych na zapoznanie się z nimi zapraszam pod adres: http://www.hollywood-mal.com/docs/html/hollywood/. Niestety nie są wszystkie opisane, wiele z nich zależy bowiem od funkcji przez nas użytej, a Andreas Falkenhahn nie jest wstanie wszystkiego sam ogarnąć. W jakimś wywiadzie przeczytałem że pisanie dokumentacji zajmuje mu więcej czasu niż sam rozwój tego języka. Mówi samo za siebie.

SetFontStyle *(style, ...)*

Poza kolorem możemy zmienić lub dodać naszej czcionce kilka rzeczy. Możemy użyć do tego stałych wbudowanych takich jak: **#NORMAL, #BOLD, #ITALIC, #UNDERLINED**. Nie będę opisywał co jaka stała robi bo pewnie łatwo się domyślić. Odnośnie tych czterech stałych stylizujących czcionkę jest takie zastrzeżenie, że każda czcionka TTF zazwyczaj ma swój wektorowy odpowiednik dla zwykłej, pogrubionej itd. I najlepiej wybrać go przy wyborze czcionki funkcjami takimi jak SetFont, OpenFont czy @FONT, raczej niż używać SetFontStyle do tego celu. Bo algorytm wbudowany w Hollywood nie będzie tak dobry jak zaprojektowana czcionka.

Poza tymi czterema stylami, mamy jeszcze **#ANTIALIAS, #EDGE i #SHADOW**. Pierwsza wygładza czcionkę, ale jedynie czcionki wektorowe - TTF, druga dodaje obwódkę. Podany kolor może być, tak jak w przypadku koloru dla czcionki, z lub bez przeźroczystości, zasady określania koloru są takie same (Zresztą nie tylko odnośnie czcionek, wszędzie tam gdzie znajduje się potrzeba podania koloru stosujemy zasady jak przy SetFontColor). #SHADOW natomiast dodaje cień pod czcionką.

W wypadku #NORMAL, #BOLD, #ITALIC, #UNDERLINED, #ANTIALIAS możemy je ze sobą łączyć w jednym wywołaniu funkcji, na przykład SetFontStyle(#BOLD|#UNDERLINED) Nie można tego robić w wypadku #EDGE i #SHADOW.

Styl **#EDGE** przyjmuje dwa argumenty kolor i grubość obrysu w pikselach, na przykład SetFontStyle(#EDGE RGB(255, 0, 0), 1) doda nam czerwony obrys o grubości jednego piksela. W przypadku **#SHADOW** wymaganych argumentów jest trzy - kolor, odległość w pikselach o czcionki rzucającej cień i kierunek rzucania cienia

```
@VERSION 4,7                                                    ; zmiana wersji na 4,7
@DISPLAY {width=800, height=600, borderless = True}             ; szerokość 800 pikseli, wysokość 600, okno bez ramki
@FONT 1, "Arial", 80, {Engine=#FONTENGINE_INBUILT}              /* Ładujemy czcionkę Arial o rozmiarze 50 i określamy sposób  *
                                                                * wyświetlania na wbudowany w Hollywood:                    *
                                                                * Engine=#FONTENGINE_INBUILT.                               *
                                                                * a nasza czcionka będzie dostępna pod identyfikatorem 1    */
@FONT 2, #SERIF, 20                                             ; Czcionka szeryfowa wbudowana w Hollywood o rozmiarze 20
@FONT 3, #BITMAP_DEFAULT, 8                                     ; Czcionka bitmapowa wbudowana w Hollywood o rozmiarze 8

UseFont(1)                                                      ;użyj czcionki o identyfikatorze 1 - w tym wypadku Arial 50 pikseli
NPrint("Cześć")                                                 ;nasz pierwszy tekst wyświetlony na ekranie funkcją NPrint

SetFontColor($FF0000)                                           ;ustawiamy w hexach kolor naszej czcionki na czerwony
NPrint("Cześć")                                                 ; i piszemy nią "Cześć"
SetFontColor(ARGB($80, $FF0000))                                ; zmieniamy przeźroczystość koloru na 50%
NPrint("Cześć")                                                 ; i sprawdzamy jak wygląda
SetFontColor(RGB (255, 0, 0))                                   ; ustawiamy kolor na czerwony ale tym razem dziesiętnie
NPrint("Cześć")                                                 ; żeby pokazać jak to robić innym sposobem
SetFontColor(ARGB(128, #RED))                                   ; zmieniamy na czerwony, półprzeźroczysty przy pomocy
NPrint("Cześć")                                                 ; stałej wbudowanej
SetFontColor(ARGB(128, $FFFFFF))                                ; Teraz powrót do białej ale przeźroczystej
NPrint("Cześć")
SetFontColor(#WHITE)                                            ; a tu pozbywamy się przeźroczystości.

SetFontStyle(#EDGE, RGB(255, 0, 0), 1)                          ; Dodajemy czerwoną obwódkę o grubości 1 piksela
SetFontStyle(#SHADOW, ARGB(128, $FFFFFF), 4, #SHDWSOUTHEAST)    ; rzucamy biały cień na dół i w prawo - SouthEast
SetFontStyle(#UNDERLINED|#ANTIALIAS)                            ; podkreślamy i wygładzamy
NPrint("Cześć")                                                 ; i co mamy - esencję kiczu :p
```

Zrzut 11

podawany tylko i wyłącznie za pomocą "kierunkowych stałych wbudowanych" (**Directional Constants**), Jest ich osiem i możecie o nich poczytać tu: http://www.hollywood-mal.com/docs/html/hollywood/PrgDirectionCst.html.

Rzućmy trochę cienia na południowy wschód, cień zazwyczaj jest przeźroczysty, więc i nasz niech będzie: **SetFontStyle(#SHADOW ARGB(128, $FFFFFF), 4, #SHDWSOUTHEAST)**. Z powodu, że nasze tło jest czarne, nasz cień będzie biały, bo czarny byłby kiepsko widoczny ;). Żeby dodać jednocześnie cień i obrys do czcionki aktualnej musimy wywołać funkcję SetFontStyle dwa razy, raz dla każdego stylu. A jeśli chcielibyśmy ją dodatkowo podkreślić i wygładzić to nawet musimy ją wywołać trzykrotnie. Zobaczcie sami na kod (Zrzut 11)

Do wyczerpania tematu czcionek pozostało nam do wspomnienia kilka funkcji, nie będę się o nich rozpisywał, możecie sami o nich poczytać na stronie Hollywooda.

GetAvailableFonts sprawdza jakie zainstalowane są czcionki na komputerze, z którego odpalany jest nasz program i zwraca nam informacje w formie tabeli zawierającą podtabelę z elementami: NAME, FAMILY, WEIGHT, SLANT, BITMAP, SIZES.

FreeGlyphCache *(mode[, id])*
Czyści z pamięci podręcznej informacje o wszystkich czcionkach albo szczególnej określonej identyfikatorem. Jeśli chcemy wyrzucić wszystkie czcionki z pamięci możemy pominąć argument ID, a jako mode podajemy 0 - zero. FreeGlyphCache(0). A jeśli chcemy wyrzucić czcionkę z identyfikatorem 2 i/lub 3, (nie używamy ich w naszym skrypcie więc pozbądźmy się ich z pamięci) to nasz rozkaz będzie wyglądał następująco: FreeGlyphCache(1, 2|3). Zwróćcie uwagę na sposób podania identyfikatorów 2 i 3, jak widać można je rozdzielać znakiem "|" i podawać dowolnie wiele.

CloseFont
Kompletnie wyrzuca z pamięci jakiekolwiek informacje o czcionce o podanym identyfikatorze.

AddFontPath
Za pomocą tej funkcji, możemy dodać miejsce w którym Hollywood będzie szukał czcionek, poza tymi domyślnymi.

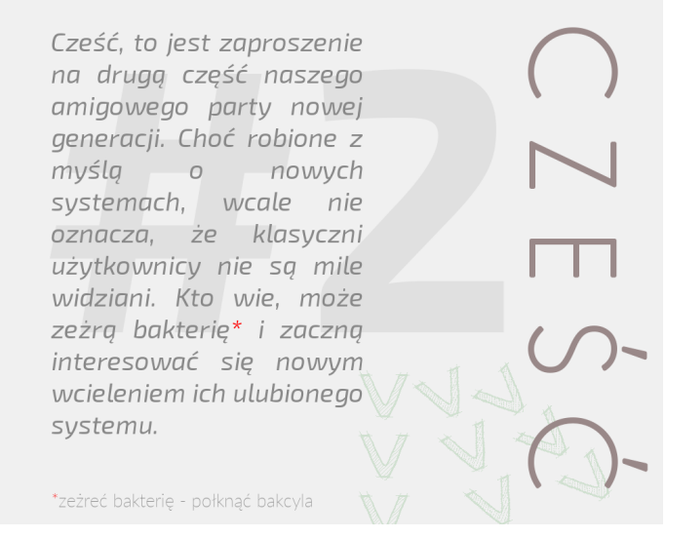
Żeby opisać wszystkie funkcje występujące w naszym skrypcie nie mogę pominąć dwóch ostatnich:

WaitLeftMouse()
Funkcja ta wstrzymuje wykonywanie skryptu dopóki użytkownik nie kliknie lewym przyciskiem myszki. W naszym wypadku potrzebna jest żeby móc cokolwiek zobaczyć, gdybyśmy jej nie zastosowali to skrypt wykonałby się z taką szybkością, że widzielibyśmy jedynie błysk.

End() Rozkaz ten kończy wykonywanie naszego programu.

Jeszcze krótko ogólnie o składni funkcji. Nawet jeśli nie przekazujemy w nawiasach żadnych argumentów, to i tak dla lepszej czytelności kodu powinniśmy je stosować.

Uffff, przebrnęliśmy jakoś przez gęste zarośla pełne literek i krojów, stylów i kolorów. Myślałem już, że to się nigdy nie skończy. Ta część może nie jest zbyt porywająca, ale od czegoś trzeba zacząć, czcionki to dobry start bo w końcu bez nich nic byśmy nie napisali. W toku pisania, mimo odgrażania się na początku artykułu, wykiełkował plan na przyszłość. Jako że pisaliśmy o czcionkach to w następnej części napiszemy o kontroli tekstu, sposobach wyświetlania, manipulowania i innych aniach.
A wjeżdżając na końcową stację, wystrzeliliśmy z pociągu na peron, bo konduktor chciał żebyśmy zabrali ze sobą te sześćdziesiąt dziewięć pustych butelek od piwa. Chyba go Bóg opuścił jeżeli myślał, że po trzech dniach i dwóch nocach ostrego obcowania z gawiedzią scenową, będziemy w stanie unieść cokolwiek innego niż nasze na wpół żywe ciała.



Konfiguracja Lightwave

waldiamiga

LightWave na Amigę wciąż jest dobrym softwarem który da się wykorzystać do rożnych mniej lub bardziej zaawansowanych zadań.
Program mimo niestandardowego GUI dość dobrze działa na MorphOSie, niestety pewnie drobnostki sprawiają że praca z Layoutem jest nieco uciążliwa szczególnie w rozdzielczościach panoramicznych.

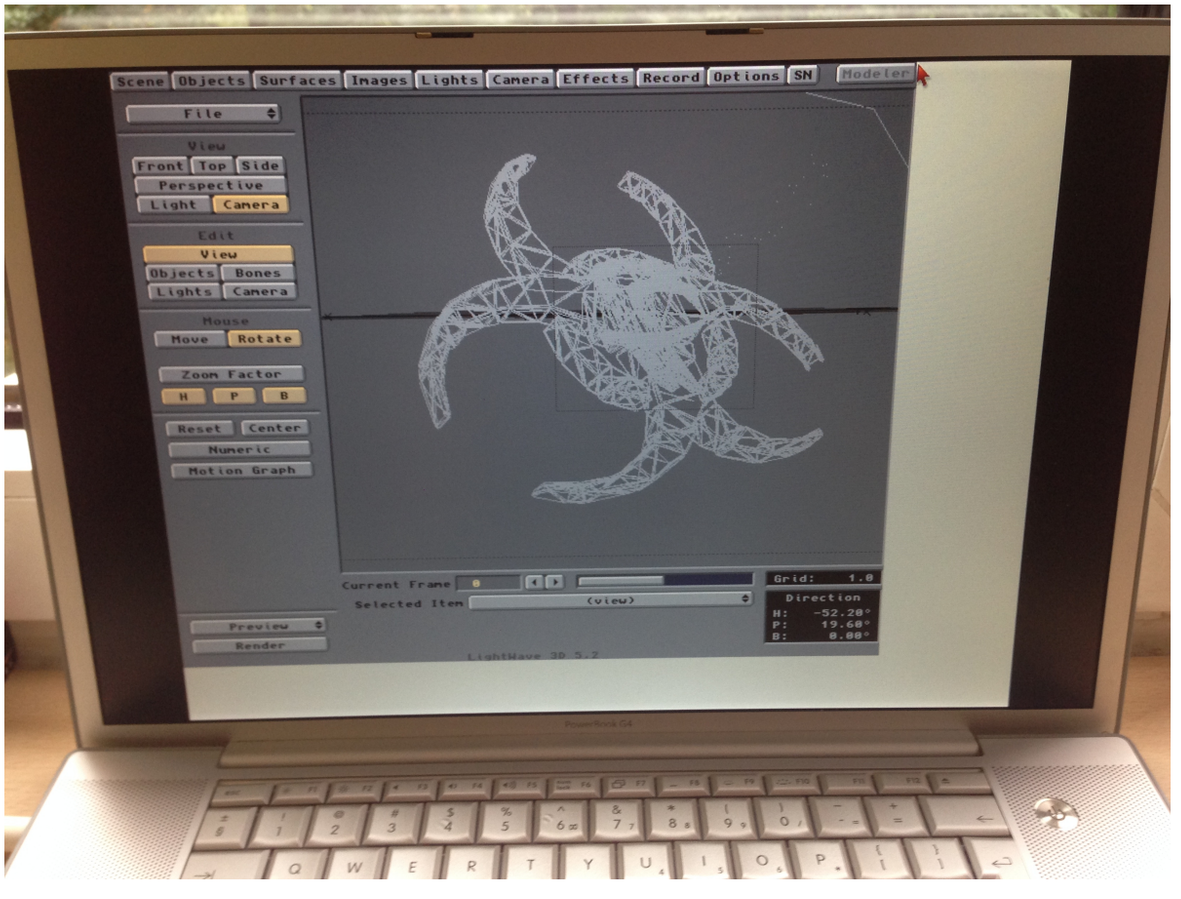
Tak to wygląda na PowerBooku G4 17" (specjalnie użyłem aparatu gdyż grabber tego nie "łapie"):

O ile w Modelerze mamy możliwość wyboru rozdziałki w której będziemy edytować obiekty, to Layout ma "sztywno" ustawione rozdzielczości obrazu, co mnie osobiście nieco "grzeje". Również renderując naszą scenę nie mamy "podglądu" wyrenderowanego obrazka.

W tym króciutkim artykule postaram się pomóc skonfigurować Layout i Modeler by wygodniej na nim pracować.
Będziemy potrzebować programu do promowania ekranów o nazwie ModePro - do pobrania na Aminecie: http://aminet.net/package/util/cdity/ModePro

Lightwave - Ja korzystam z wersji 5.2, chyba ostatniej która została wydana na Amigę.

Promotor ekranu instalujemy (instalator lubi się pogubić), wrzucamy sami do WBStartup, kto jak woli, ja preferuje tą ostatnią metodę, plik konfiguracyjny do Prefs. Korzystałem z lokalizacji w języku Polskim.

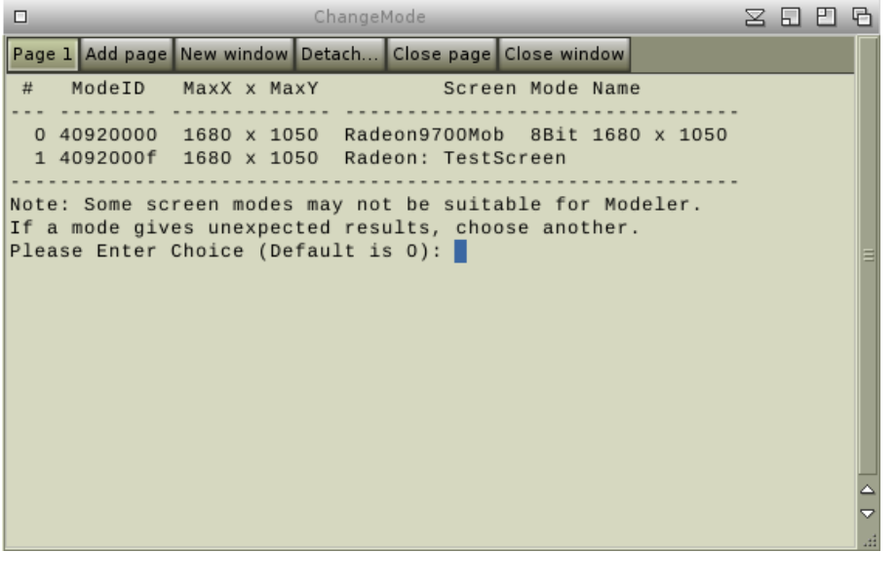


Konfiguracja Lightwave:

Uruchamiamy programik o nazwie CyberGFXRendererPrefs, pchełka ta leży sobie w katalogu Toaster:Utilities.
Ustawiamy pożądaną przez nas rozdzielczość, oczywiście najlepiej natywną dla naszego monitora czy matrycy, w moim przypadku jest to 1680x1050. Dajemy Save.

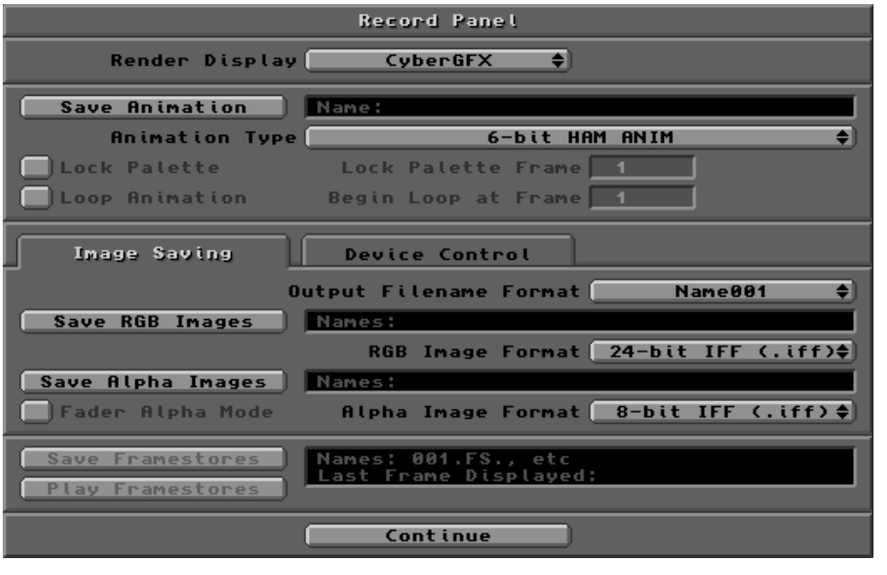
CyberGFXRendererPrefs

Następnie Modeler.
Odpalamy program ChangeMode, znajdziemy go w Toaster:Utilities.
Po wskazaniu pliku MOD-Config (Toaster:LightWave_Support/MOD--Config) ustawiamy docelową rozdzielczość dla Modelera.



ChangeMode

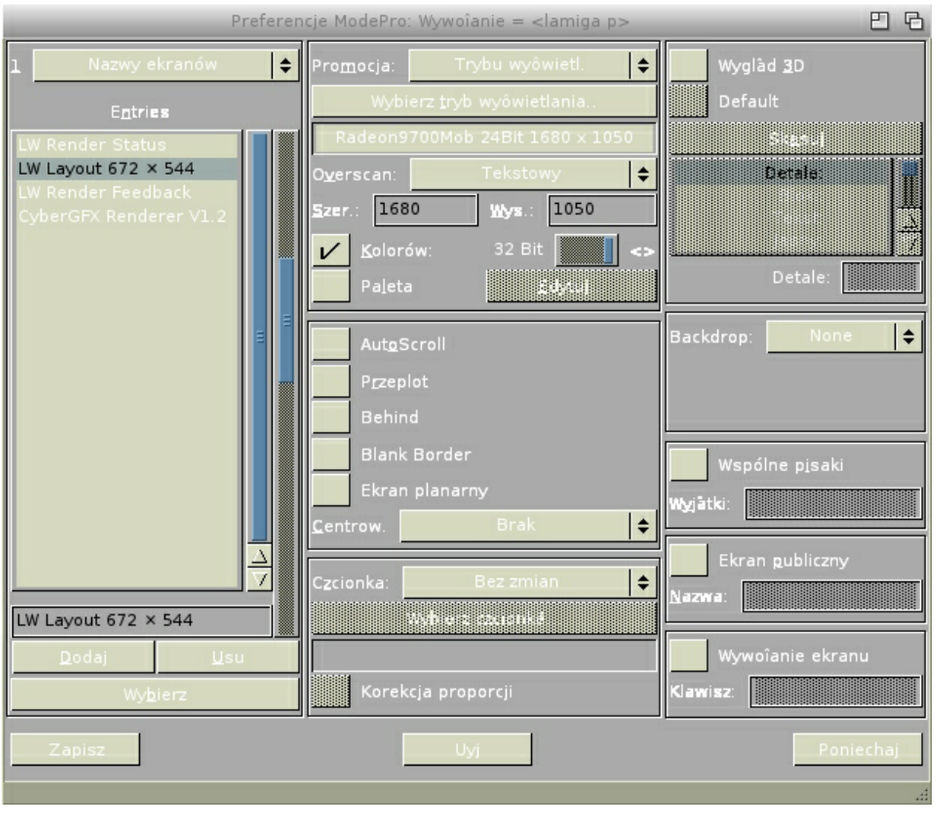
Czas na dołączenie pluginu CyberGFXRenderer do LightWave. W tym celu startujemy Layouta (odpali nam się najprawdopodobniej w kulawej rozdzielczości, ale dojdziemy i do tego), klikamy zakładka "Options", potem "Add Plug-ins" i wyszukujemy pliku o nazwie CyberGFXRenderer.p, zwykle znajduje się on w Toaster:Plugins/Layout.
Potem klikamy na zakładce "Record", tam w "Render Display" ustawiamy CyberGFX, tak jak na zdjęciu. Zamykamy LightWave.



Dołączanie CyberGFXRenderer

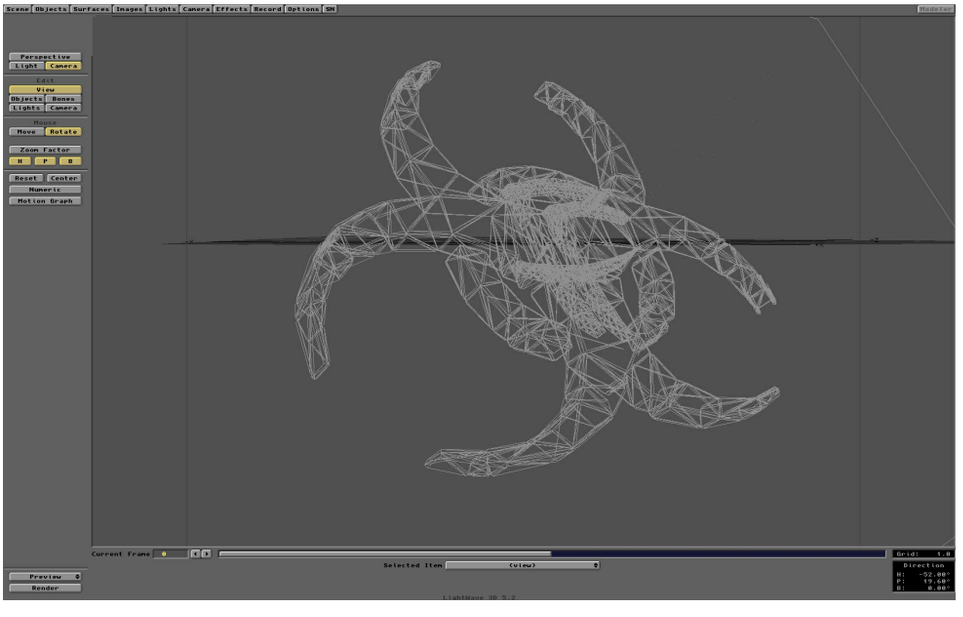
No to czas na konfiguracje ekranów w promotorze ModePro.

W SYS:Prefs klikamy ModeProPrefs lub naciskamy kombinacje klawiszy Amiga (Jabłko w przypadku maszyn Apple) +P.
Pierwsza zakładka - ustawiamy na Nazwy Ekranów, później klik na dodaj, dalej wybierz i na requesterze wybieramy "LW Layout 672 x 544". Zakładka "promocja" ustawiamy na “Tryb wyświetl", ustawiamy odpowiedni dla nas tryb wyświetlania, overscan dajemy na "Tekstowy", zaptaszkowujemy "Kolorów" i slider przesuwamy zgodne z wybranym trybem (24--32bit) który wybraliśmy wcześniej. Tę procedurę powtarzamy dla wszystkich wpisów czyli CyberGFX Renderer, LW Render Status oraz Feedback jednak przed każdym trzeba kliknąć "dodaj". Powinno to wyglądać mniej,więcej jak na zdjęciu obok



Ponownie odpalamy LightWave, teraz obraz powinien być jak żyleta w rozdziałce którą mu wskazaliśmy. Dzięki pluginowi CyberGFX powinniśmy również zobaczyć efekt naszej pracy.
Niestety, nie zobaczymy (a raczej będziemy widzieć black screen) ekranu liczonych mozolnie segmentów czy "lizania" antialiasingiem, ale to jedyna wada tego rozwiązania.

Mam nadzieję że ten krótki tekst nieco ułatwi Wam pracę czy zabawę w LightWave na MorphOSie.



Kotek na motylku

waldiamiga

Lubię sobie poemulować, szczególnie komputery i konsole których na oczy nie widziałem a tym bardziej nie używałem.
Konsole Atari Jaguar widziałem, ale nigdy na niej nie grałem. Czas nadrobić te zaległości, a po tym tekście mam nadzieję, że wy też choć spróbujecie.

Czego mamy szukać...

Emulatora Virtual Jaguar - niestety oparty na źródłach bardzo starej wersji 1.0.7 z końca... 2003 roku (nowsze wymagają środowiska QT),

Co ciekawe został on przeportowany w dwóch wersjach przez dwóch różnych programistów. Pierwszy z nich, Maksim Ilin, pracuje nad tą emulką już od dłuższego czasu. Pierwsza wersja pojawiła się już kilka lat temu, niestety nie wiedzieć czemu jego port u mnie nie działa, więc z automatu został mi soft znanego w naszym światku Faba.
Możemy go znaleźć pod tym adresem:
http://chris.highlander.free.fr/VirtualJaguar1.0.7_Fab1.lha

Emulator wykorzystuje biblioteki SDL, co jest jego dużą wadą - mimo bezczynności mam 100% zajętości procesora.

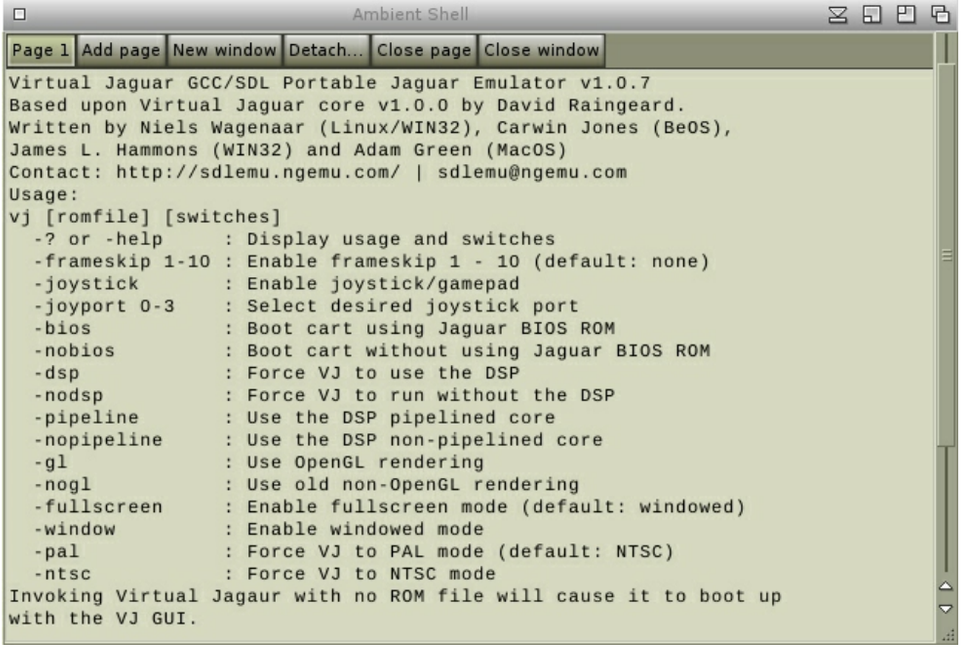
Co dalej...

Bios Jaguara, plik jagboot.rom nie jest dołączony do archiwum więc musimy go poszukać w sieci, nadmienię że nie każda gra wymaga biosu.
Gry - Virtual Jaguar uruchamia tylko obrazy gier zapisanych na kartridżach, nie ma możliwości korzystania z przystawki Jaguar CD co jest kolejną wadą.

Czas odpalać...

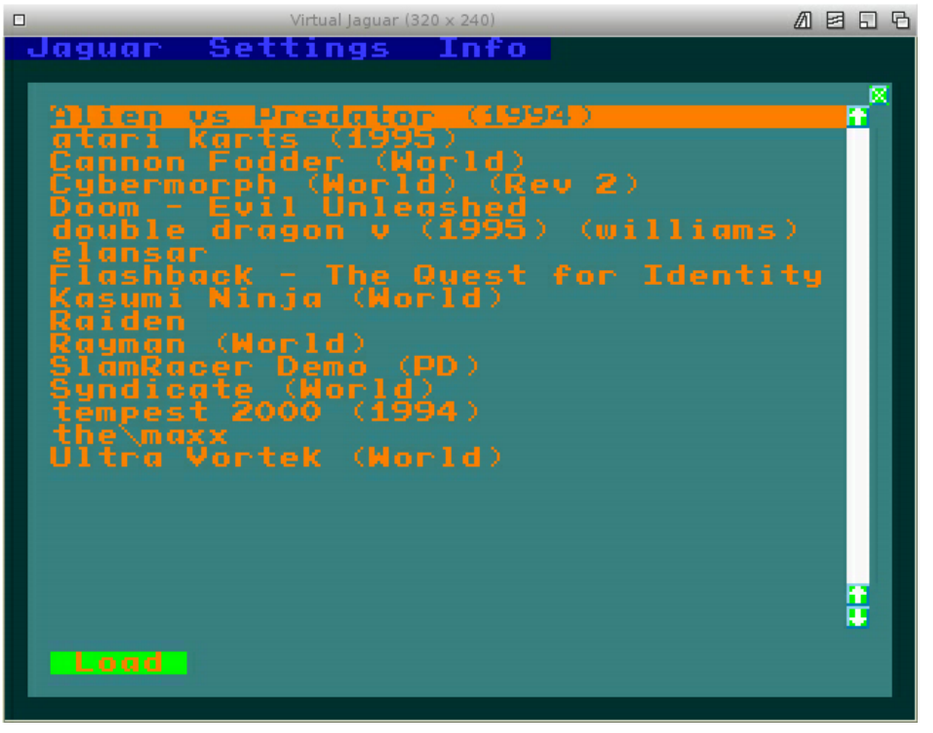
Emulator możemy uruchomić w dwojaki sposób: z Shella gdzie mamy sporo parametrów emulacji, lub tradycyjnym dwuklikiem.

Emulowany Jaguar uruchamia się standardowo w niezbyt urodziwym okienku, jest też proste menu w którym część opcji (Audio i Video) nie działa.



Emulator uruchomiony poprzez Shell

Najlepiej edytować plik konfiguracyjny vj.cfg i dostosować go do naszych potrzeb. Emulator ponoć wykorzystuje joysticki, jednak nie mogłem tego sprawdzić.



Emulator w standardowym trybie uruchomienia

Co działa...

Z powodzeniem uruchomiłem kilkanaście gier, oczywiście znalazły się "czarne ofce" które nie są kompatybilne z emulatorem.
Ale najważniejsze gierki, takie jak Alien Vs. Predator, Ultra Vortek, Kasumi Ninja, czy graficzny majstersztyk pt. Rayman.

Gra się fajnie, AvP wciągnął mnie jak odkurzacz Zelmera.
Córka chwilę grała w Raymana - strasznie jej się podobała.

Działają też znane nam z klasycznej Amigi Flashback, Cannon Fooder, Syndicate.
Istnieje lista kompatybilnych gier z emulatorem, ale obawiam się że dotyczy ona nowszych wersji emulki.

Jak działa...

Testowałem emulator na PowerBooku G4 1,67GHz i przy włączonym DSP... nie da się grać. Dźwięk szarpie, obraz ma może z 5 klatek na sekundę. Masakra!

Nie wiem czym to jest spowodowane, być może słabym kodem emulatora lub DSP, zastosowaniem bibliotek SDL?
Może potwór z G5 2,7GHz ogarnie emulacje na zadowalającym poziomie.

Sytuację znacznie poprawia wyłączenie owego DSP, wtedy praktycznie każdy szpil jest grywalny, jednak mogą pojawić się problemy np. Alien Vs. Predator wiesza się przy wejściu do windy.

Reklama Jaguara z ACS (1995 r.)

Zamiast zakończenia - podsumowanie na "plusy i minusy":

+ Alien Vs. Predator.

+ Że w ogóle coś działa :)

+ Trochę Amigowej klasyki na wirtualnych kartridżach

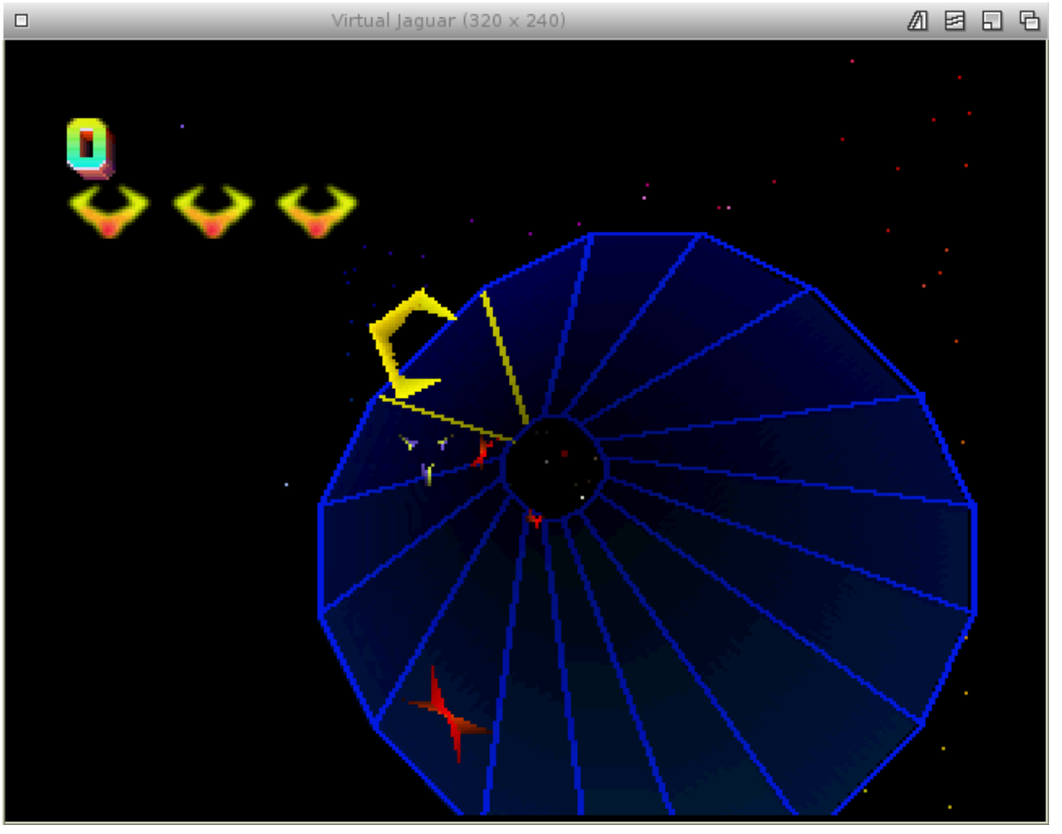
- SDL czyli 100% zajętości procesora.

- Niedziałające opcje w menu

- Mega wymagania sprzętowe w przypadku użycia DSP.

AVP (Alien versus Predator)

Rayman



Tempest 2000

Cannon Fodder

Mapparium

Don Rafito

Wakacje; czas odpoczynku i rekreacji. Czas ładowania baterii, pochłaniania przyjemnych dla ducha i ciała wrażeń i doznań. Uprawianie sportów to jedna z form wakacyjnego wypoczynku. Letnia pogoda zachęca m.in. do podróży, biegania, czy organizowania wycieczek rowerowych.

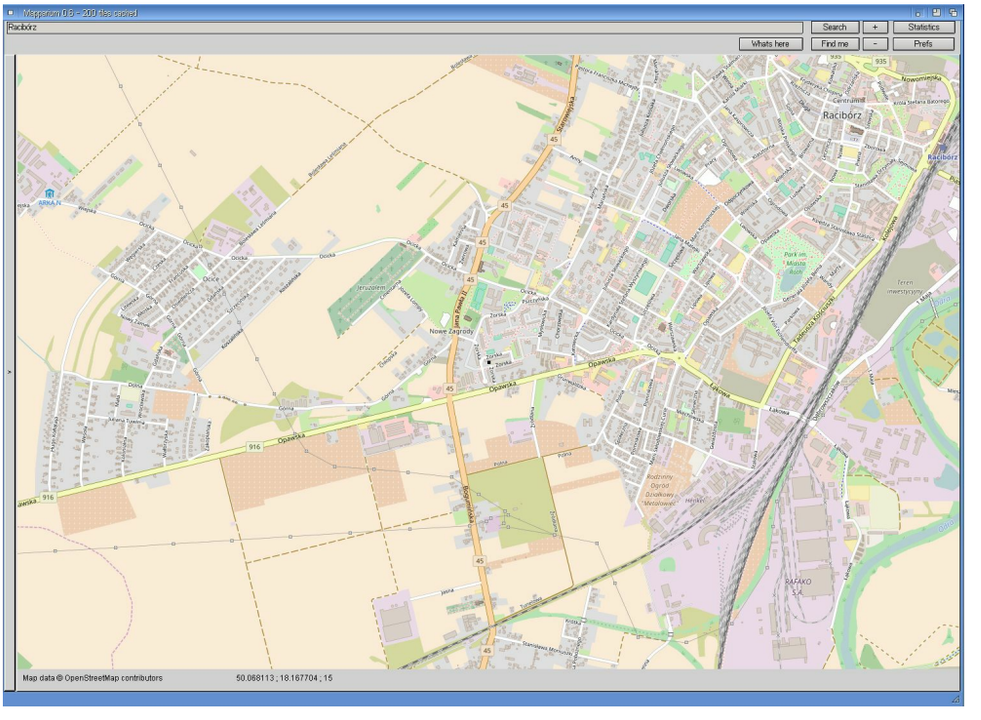
Tak się składa, że rozwój technologii i informatyzacji, wkradł się także i tu, poprzez coraz powszechniejsze smartfony z GPS-em, aplikacje do pomiaru aktywności i publikację swoich osiągnięć na portalach społecznościowych. Na jedno to dobre, gdyż działa motywująco; chcemy być coraz ładniejsi, lepsi, szybsi, sprawniejsi. I chcemy się tym chwalić, nawet jeśli czasami gdzieś w środku jesteśmy introwertykami. Na drugie; chyba nie do końca zdajemy sobie sprawę z tego, że wszystkie te dodatkowe elementy, które towarzyszą nam np. podczas zwykłej przejażdżki przez las, trochę okradają nas z prywatności. No ale chyba nie da się tego procesu zatrzymać i trzeba się pogodzić z tą domeną naszych czasów.

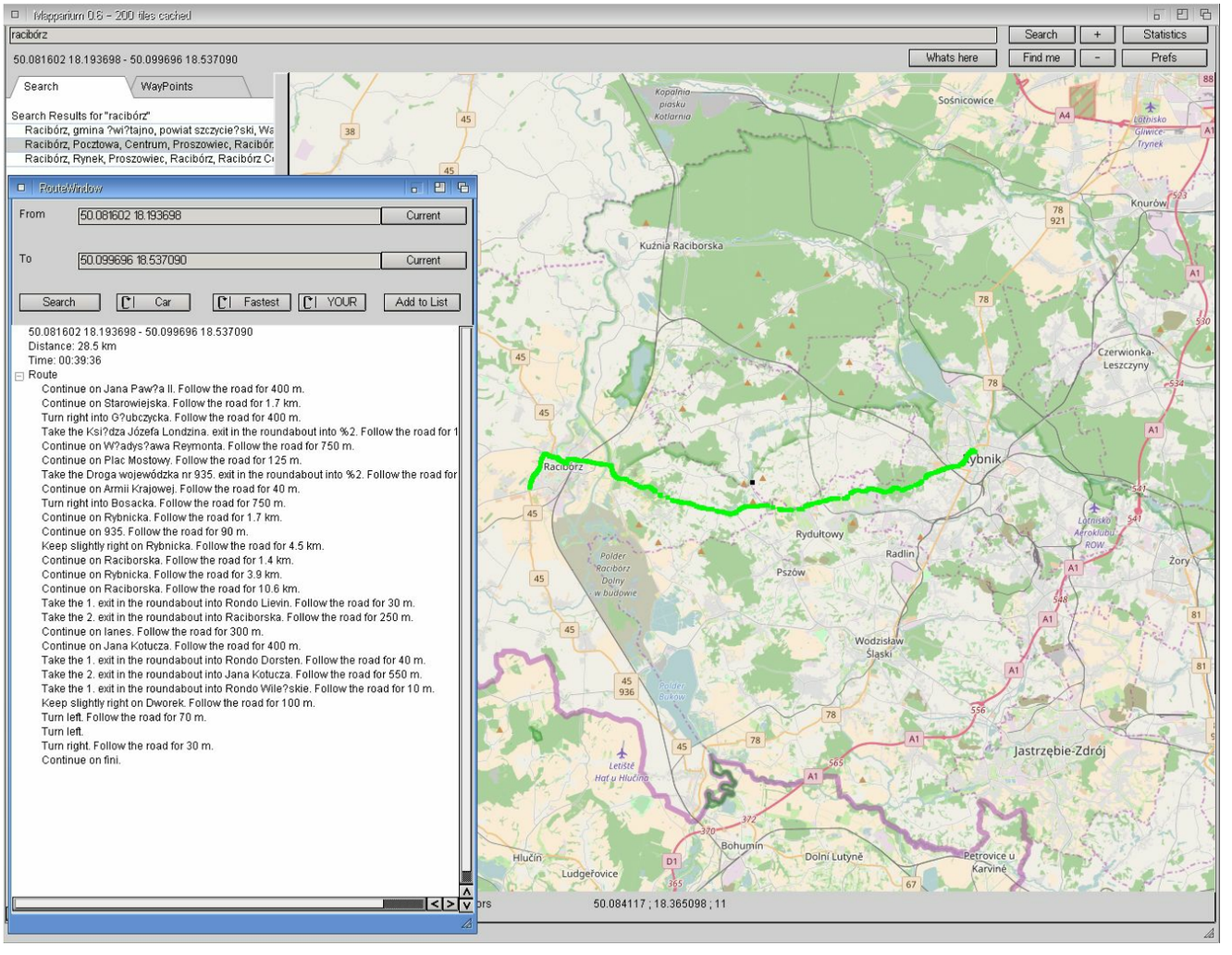
Dobra, skoro się już na to godzimy, to warto by było zaprezentować narzędzia, które pomogą nam np. zapoznać się z przebiegiem trasy owej wycieczki rowerowej, czy zwyczajnie; zaplanować wyjazd samochodem.

Na rynku jest mnóstwo takich rozwiązań; począwszy od darmowych map obsługiwanych czy to z poziomu przeglądarki na komputerze, czy na urządzeniu mobilnym, skończywszy na systemach, gdzie trzeba co jakiś czas wykupywać subskrypcję. Którą wybrać? Każdemu odpowiada co innego. Tak czy siak, chyba nie ma rozwiązania, które by miało absolutnie wszystko.

Jednym z takich, moim zdaniem dość dokładnym wyborem, jest projekt *Open Street Map*, oraz przeglądarka dla systemu Aros obsługująca go, o nazwie *Mapparium*. Sam *OSM*, zapoczątkowany 1 lipca 2004 roku przez **Steve'a Coasta**, jest inicjatywą społeczną, opartą na ideologii *Wikipedii*; zarejestrowani użytkownicy sami nanoszą poprawki i edytują mapę. Natomiast prezentowany program stworzony został przez **Marcusa "ALB42" Sackrowa**. Prócz Aros'a i386 i ARM, autor opracował także wersję dla MOS'a, AOS 4.x (starsza wersja) i Amigi Klasycznej.

Przyznam szczerze, że nie znałem do końca systemu map od *OSM*, choć troszkę o nim słyszałem i po odpaleniu prezentowanej przeglądarki, to co mnie w niej urzekło, to dokładność. Mnóstwo szczegółów; drogi, instytucje,





adresy, trasy piesze i rowerowe, place zabaw, nawet można znaleźć jakieś prowizoryczne ścieżki wydeptane przez przechodniów. Sama przeglądarka dość dobrze radziła sobie z wczytywaniem mapy ze serwisu i nie było większych problemów. Wyszukiwanie miejsc czasami sprawiało małe problemy; żeby coś znaleźć w *Mapparium*, jakiś obiekt, adres czy instytucję, trzeba było mieć przynajmniej znikomą wiedzę na temat szukanego tematu. Wyznaczanie trasy nie jest niestety do końca intuicyjne. Najpierw trzeba znaleźć żądny punkt początkowy, wczytać jego dane geograficzne, następnie punkt docelowy i postąpić jw., a dopiero *Mapparium* wyznaczy trasę wraz z opisem... w większości po angielsku. To też jest jakieś utrudnienie, gdyż dla osób nieznających dobrze angielskiego może to sprawić problem. Sam schemat przebiegu podróży jest dość dobrze oznaczony, a dla wprawionego podróżnika, nawet amatora, trasa wyznaczona przez program będzie raczej atutem. Niestety, próby znalezienia alternatywnej drogi, czy manipulowanie przy planowaniu trasy przebiegały dość opornie, wliczając w to zawieszenie przeglądarki i całego systemu.

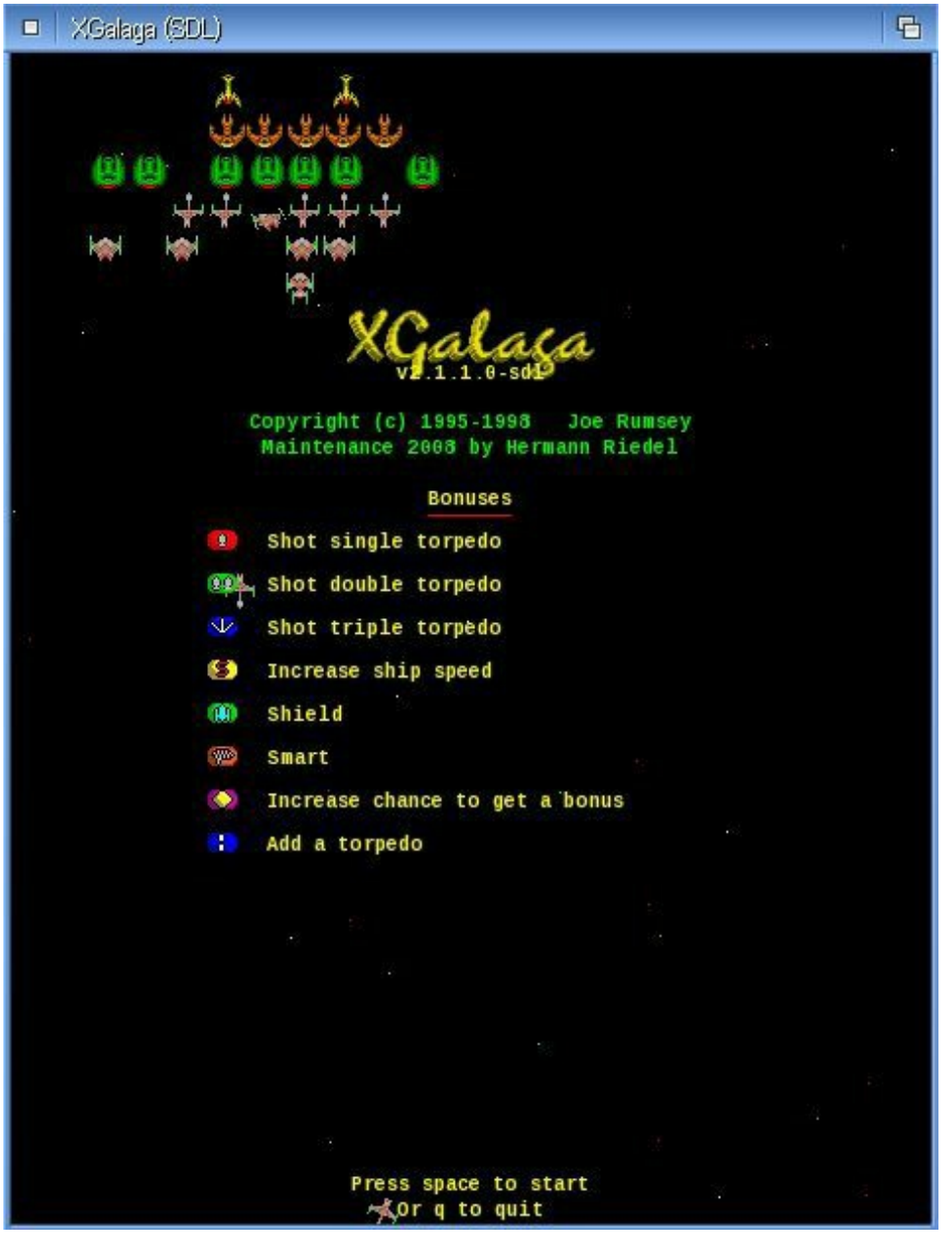
Reasumując. Dokładność mapy oferowana przez *OSM*, w połączeniu z *Mapparium*, przenosi użytkownika w czasy, kiedy to na podwórku studiowało się z kolegami mapę samochodową, „pożyczoną" z samochodu ojca. Zapoznanie się z dokładną topografią i względne wyszukiwanie miejsc, to dość silne atuty programu. Pozostałe funkcje wymagają jeszcze dopracowania. Myślę, że prędzej czy później to nastąpi. Po wpisach autora na swoim blogu można wnioskować, że „work in progress". Was na razie zachęcam do wypróbowania prezentowanej propozycji i zapoznania się z systemem *Open Steet Map*.

Tytuł: Mapparium
Platforma: Aros i386, Aros ARM, MOS, AOS4.x, m68k
Firma: Marcus "ALB42" Sackrow
Rok produkcji: 2016
Ocena: 6

XGalaga

Don Rafito

Czasami to, co wydaje się oczywiste, jest zawiłe i zagmatwane. Tak samo jest z grami komputerowymi; szukamy czasami na siłę wielkich hitów, rozczarowując się nimi a mamy pod ręką sprawdzone tytuły, które przez wiele lat sprawiały nam wiele frajdy.



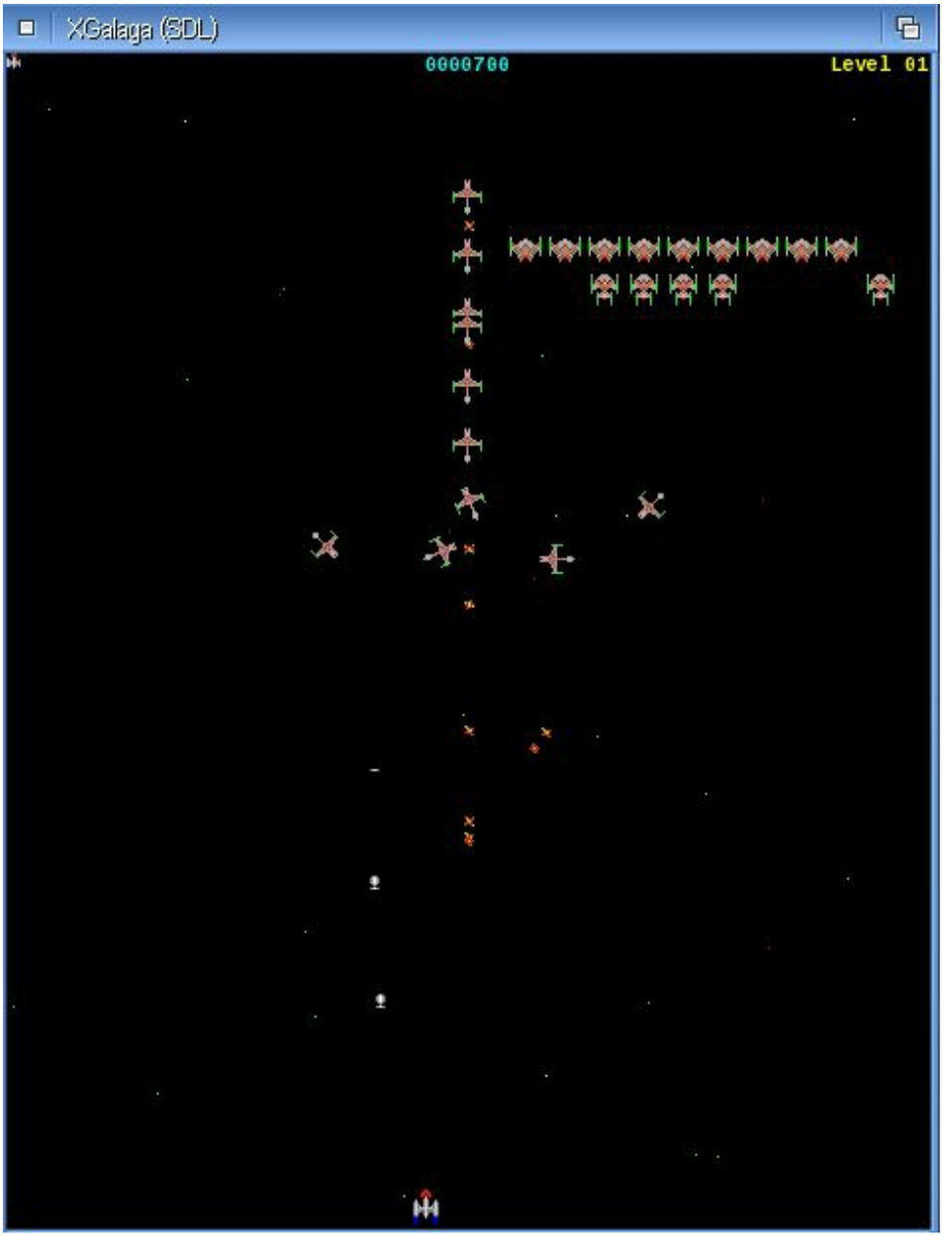
Jednym z takich sprawdzonych tytułów jest kosmiczna strzelanka o nazwie *XGalaga*. Jak sama nazwa wskazuje, propozycja należny do dość bogatej już rodziny, odmian gry *Galaxa*. Żeby nie powiedzieć, że jest jej kolejnym klonem. Fabuła znana. Nasz malutki stateczek i hordy najeźdźców. Nasza zwinność i spostrzegawczość, jak przystało na wyżej wspomnianą strzelankę kosmiczną, będącą także po części zręcznościówką, kontra zastępy wrogich i uzbrojonych statków.

Każda z odmian oryginału, zazwyczaj zawiera w sobie mniej, lub bardziej przydatne dodatki. W prezentowanym tytule zawarto np. system bonusów; podczas ferworu walki zauważymy, że od czasu do czasu pojawi się spadający klocek zawierający cenny ładunek. Jeśli uda nam się go przechwycić, możemy wzbogacić się w rozszerzenie uzbrojenia, powiększyć siłę ognia, czy uzyskać dodatkowe osłony. Fajnym dodatkiem jest także możliwość wstawienia parametru przed uruchomieniem gry za pomocą linii poleceń w Shell. Do ciekawszych opcji należy: odpalenie gry w oknie (możliwość taka istnieje także podczas standardowego uruchomienia gry, poprzez naciśnięcie klawiszów **ALT+ENTER**), uruchomienie gry w oknie o zadanych parametrach wielkości (domyślnie to 468 × 596),

uruchomienie gry bez dźwięku, czy wydruk tabeli osiągnięć.
Skróty klawiszowe to kolejny atut *XGalagi*;
P-pauza, **S-wyłączenie dźwięku**, czy **Q-wyjście** bezpośrednio do systemu.

Sterowanie naszym statkiem odbywa się za pomocą klawiszy kursorów i spacji, ale uwaga, dodano także obsługę lowlevel.library i *Posejdona*. Tak, więc jeśli w komputerze mamy wpięty jakikolwiek joystick USB, bez problemu możemy go używać podczas masakrowania wrogich jednostek.
Strona graficzna ewidentnie nawiązuje do oryginału. Obiekty oraz ich animacja, są na dobrym poziomie, choć na kolana nie powalają. Dużym plusem byłoby tu wstawienie lepszego tła otchłani kosmosu, no. Z widokiem na jakieś planety czy galaktykę.

Dość realistyczne efekty dźwiękowe, kiepsko sprawdzają się ze stroną graficzną, która jak wspomniałem wyżej bliższa jest reto-klimatom. Nie, żebym się czepiał jakości sampli, bo te są dość niezłe, ale albo się trzymamy oryginału, albo robimy grę na miarę aros'owych możliwości sprzętowych. Drugim mankamentem, jakie mnie razi, to brak dynamicznego podkładu muzycznego. Strzały i wybuchy po jakimś czasie zaczynają się nudzić i dynamika rozgrywki zaczyna spadać. No dobra, to w takim razie odpalmy sobie jakąś mp3-jkę z playera. I tu jest kolejny problem; jeśli zapuścimy sobie ulubiony kawałek, to żegnamy się z odgłosami walki, gdyż gra nie do końca poprawnie współpracuje z *AHI* i zwyczajnie milknie.



Fani wszelkich *Galaxian'ów* znajdą w tytule coś dla siebie. Grywalność jest na dość przyzwoitym poziomie. Możliwość łapania bonusów i uruchomienia gry z dodatkowymi parametrami na pewno przyćmi nieco braki w stronie graficznej i brak pełnej koordynacji ze stroną dźwiękową prezentowanego tytułu.
I choćby z tego względu zachęcam Was, do zapoznania się z strzelanką.

Tytuł: XGalaga
Platforma: Aros i386
Firma: Joe Rumsey
Rok produkcji: 2012
Grafika: 7
Muzyka: 5
Grywalność:8

Pomidor

truman burbank

Podstawowym problemem podczas pracy z komputerem jest wydajność. Cała sztuka polega na tym żeby czas wykorzystać optymalnie.

Idealnie byłoby żeby jak najwięcej czasu poza pracą zostało na prawdziwe życie. Maksymalne natężenie umysłu i pośpiech nie jest właściwą drogą - człowiek może być wydajny przez określony czas, potem następuje zmęczenie.

Dlatego już od wielu lat pisze się o tym jak pracować przy komputerze jak najmniej obciążając organizm. Problemy ze wzrokiem, kręgosłupem, zespół cieśni nadgarstka - to nie są fajne sprawy, a taka jest cena pracy w jednakowej pozycji przez wiele godzin.
Zaleceń jest dużo. Najpierwsze jakie pojawiły się dotyczą ergonomii - odpowiedniej wysokości biurka, krzesła, odległości oczu od monitora, i tak dalej. Pewnie każdy z was przynajmniej raz zetknął się z artykułem poruszającym taki temat.

Dziś jednak chcę napisać o czymś troszkę innym. O optymalizacji samego "posiedzenia".
Użytkownicy PC (Windows, Linux) powinni znać program Workrave. Jego założeniem jest relaksacja przy komputerze, ale nie odbywa się to przez kojące dźwięki. To program proponujący w czasie pracy pauzy - pokazane są ćwiczenia które można wykonać przy biurku. Istotnym elementem jest czasomierz, który proponuje nam mikroprzerwy (na odpoczynek dla oczu) i przerwy (na proste ćwiczenia). Wymaga on indywidualnej konfiguracji, trzeba go dostosować do własnego rytmu pracy, po to by pomagał a nie denerwował.

Idea jest słuszna, i sam program jest w porządku. Ale nie rozwiązuje istotnego problemu. W czasie pracy mamy możliwość robienia czegoś w tle - jest multitasking, jest moc komputera - czemu nie. Pół biedy jeśli są to "zbożne" czynności. O wiele częściej jednak niestety rozpraszamy się - facebook, gra online w przeglądarce. Zdarza się też że robimy kilka rzeczy naraz i w rezultacie żadna nie jest skończona.

W tym miejscu z pomocą przychodzi technika "pomodoro" (z włoskiego - pomidor). Opracowana została końcem lat 80-tych przez Francesco Cirillo. Nazwa wzięła się od minutnika kuchennego w kształcie pomidora, którego używał Cirillo jako student uniwersytetu. Czas dzielony jest na "pomidory" czyli okresy 25-minutowe.
Istotą jest skupienie się na jednej czynności.

Oryginalnie procedura była rozbita na sześć kroków:

1. określenie zadania do wykonania
2. obiecanie sobie że przez kolejne 25 minut skupię się na tym zadaniu i tylko na tym
3. jeżeli w czasie tych 25 minut przyjdzie ci do głowy że masz coś innego ważnego do wykonania - zapisz to na kartce
4. kiedy pomidor zadzwoni zaznacz zadanie na kartce jako wykonane
5. zrób sobie krótką przerwę - relaks (oddychanie, medytacja), kawka, krótki spacer; coś co wyłączy twój mózg, to ma być czynność niezwiązaną z pracą
6. po pomyślnym zaliczeniu czterech pomidorów - zrób sobie dłuższą przerwę - 20 minut; może nawet 30 - w tym czasie mózg przyswoi sobie nowe informacje i odpocznie przed kolejnymi zadaniami

Taki schemat postępowania jest przedstawiony na oficjalnej stronie https://cirillocompany.de

Tak, zgadza się, na wszystkim można zarobić. Też nie przypuszczałbym że na przedstawienie w sumie prostej metody można przeznaczyć książkę, audiobooka, czy wogóle zbudować na tym firmę. Żeby było śmiesznie, "pomodoro" nie ma nawet własnego hasła we włoskiej Wikipedii.

Technika "pomodoro" doczekała się modyfikacji. Było to konieczne, na przykład są zadania które wymagają więcej czasu niż 25 minut, a są i takie które zrobimy w 10 minut. Co robić w takich sytuacjach?
Ważne żeby w pomidorach nie było pustych miejsc, i żeby tak podzielić zadania by je móc realizować.

Czyli do małego zadania dorzucamy kolejne, a większe zadania dzielimy na kilka pomidorów. Jest też możliwość zmiany tych 25 minut na inny czas. Ważne jest żeby trzymać się pierwotnej koncepcji - skupieniu się na jednym zadaniu, i rytmie (zadania i przerwy).

Warto zadbać przed pracą żeby w trakcie zadań nic nas nie rozpraszało (np. telefon), czy wogóle nie było potrzeby wstawania od biurka, czy odrywania się w jakikolwiek sposób.

Oczywiście kim byłby współczesny człowiek, gdyby nie wymyślił sobie programów zamiast minutnika.

Takim dosyć znanym (na Windows i Mac) programem jest Tomighty. Czy są podobne na Amigę - nie wiem, wątpię. Ale też i moc naszych komputerów, i pewne ich braki, pozwalają łatwiej skupić się na tym co robimy.

Próbowałem wymienionych przez siebie wyżej programów (zarówno Workrave jak i Tomighty). Zostaję przy tradycyjnym pomidorze, za około dziesięć złotych.
Wydaje się mi, że programy dodatkowo rozpraszają - a do ciachania minutnika idzie się przyzwyczaić.

REKLAMA

W następnym numerze

Co będzie w kolejnym numerze tego nie wie nikt. Nie jest uformowany stały skład (redakcja pisma), ani nie ma zapasu tekstów.

Bardzo możliwe że przyszły numer będzie zdominowany przez AROS-a - system z długimi tradycjami, rozwijany bez hałasu i konsekwentnie.

Na pewno będzie ciąg dalszy poznawania programu Hollywood, prawdopodobnie gry pod MorphOS.

Tematyka zależy od tekstów jakie napłyną, żaden system nie jest faworyzowany i nic nie jest przesądzone.
Na pewno będziemy zwiększać objętość i rozwijać się razem z wami.

AmigaOne

The most powerful AmigaOS Computer ever

X5000

PowerPC

64-bit

dual-core

2.0GHz

For more details please visit:

www.a-eon.com/x5000

(or your participating local Amiga dealer)

PRE ORDER NOW AVAILABLE

CLOSE ENCOUNTER SYSTEM

prices starting from £1699 inc. AmigaOS 4.1 (£1415 Ex VAT)

AmigaOne, AmigaOS and the Boing Ball are trademarks of Hyperion Entertainment used under License

A-EON